Documentation

OpenStage 60/80 - XML Applications

Developer's Guide

A31003-S2000-R100-7-7620

Communication for the open minded

Siemens Enterprise Communications
www.siemens.com/open

SIEMENS

Communication for the open minded

Siemens Enterprise Communications
www.siemens.com/open

Copyright © Siemens Enterprise Communications GmbH & Co. KG 2004
Hofmannstr. 51, D-80200 München

Siemens Enterprise Communications GmbH & Co. KG is a Trademark Licensee of Siemens AG

Reference No.: A31003-S2000-R100-7-7620

The information provided in this document contains merely general descriptions or characteristics of performance which in case of actual use do not always apply as described or which may change as a result of further development of the products. An obligation to provide the respective characteristics shall only exist if expressly agreed in the terms of contract. Availability and technical specifications are subject to change without notice.
OpenScape, OpenStage and HiPath are registered trademarks of Siemens Enterprise Communications GmbH & Co. KG.
All other company, brand, product and service names are trademarks or registered trademarks of their respective holders.

1 Introduction 2
1.1 Overview 2
1.2 Prerequisites 3
2 Operating Mode and Features 4
2.1 Infrastructure 4
2.2 Basic Operating Mode 5
2.3 Push Mode 6
2.4 Cache 6
3 Setup and Program Start 8
3.1 Configuring an Application on the Phone 8
3.2 Setting a Proxy 13
3.3 Reconfigure and Remove an Application 15
3.4 Starting the Application 18
3.5 Mobile Users And Privacy 19
4 XML Object Reference 20
4.1 Encoding/Internationalization 21
4.2 Phone 21
4.3 Display 23
4.4 Screen 25
4.5 Action 27
4.6 Command 30
4.7 Direct Keypad Input 35
4.8 Alert 44
4.9 List 49
4.10 TextBox 54
4.11 Ticker 59
4.12 Hidden 61
4.13 Form 63
4.14 Image 96
4.15 Player 98
4.16 Phone Number 105
5 Push Capability 106
5.1 The 'Push' Request 109
6 Appendix 112
6.1 Glossary 112
6.2 XML Schema 113

1 Introduction

1.1 Overview

With the OpenStage 60/80 models, a multitude of applications with GUI can be deployed.

As XML and HTTP serve as client/server interface, the technologies commonly used in web applications can be used: Java Servlets, JSP, PHP, CGI etc., delivered by servers such as Tomcat, Apache, Microsoft IIS.

The Push feature enables the server-side application to send data to the phone in an unsolicited way.

The XML application interface is available in the SIP firmware (from V1R3.x) or the HFA firmware (from V2R0.x) for OpenStage 60/80 phones.

This document describes the XML interface and the error handling in case the XML data is invalid. Please note that the error handling is not visible to the user. If, for instance, some elements of an object are missing, the remaining objects are displayed anyhow. It is recommended to validate the XML documents using the XML schema provided here (see Section 6.2, "XML Schema").

This document is aimed at Siemens and Third Party developers and covers the following areas:

- Operating Mode and Features
- Setup and Program Start
- XML Object Reference
- Push Capability
- Appendix

1.2 Prerequisites

For developing applications for OpenStage phones, the following components and tools are required or recommended:

1. OpenStage 60/80 SIP phone with appropriate firmware version (V1R3.x or higher).
2. Infrastructure:
   - Connection to an IP network.
   - Web browser to configure the application on the phone using the Web Based Management. Alternatively, the phone's local admin menu can be used.
   - Web server for the delivery of XML data, like Tomcat, Apache, Microsoft IIS. Please note that the MIME type for the XML documents has to be set to `text/xml` on the server.
   - Application platform running on the server, for instance Java Servlet, ASP, CGI, PHP.
3. XML Editor:

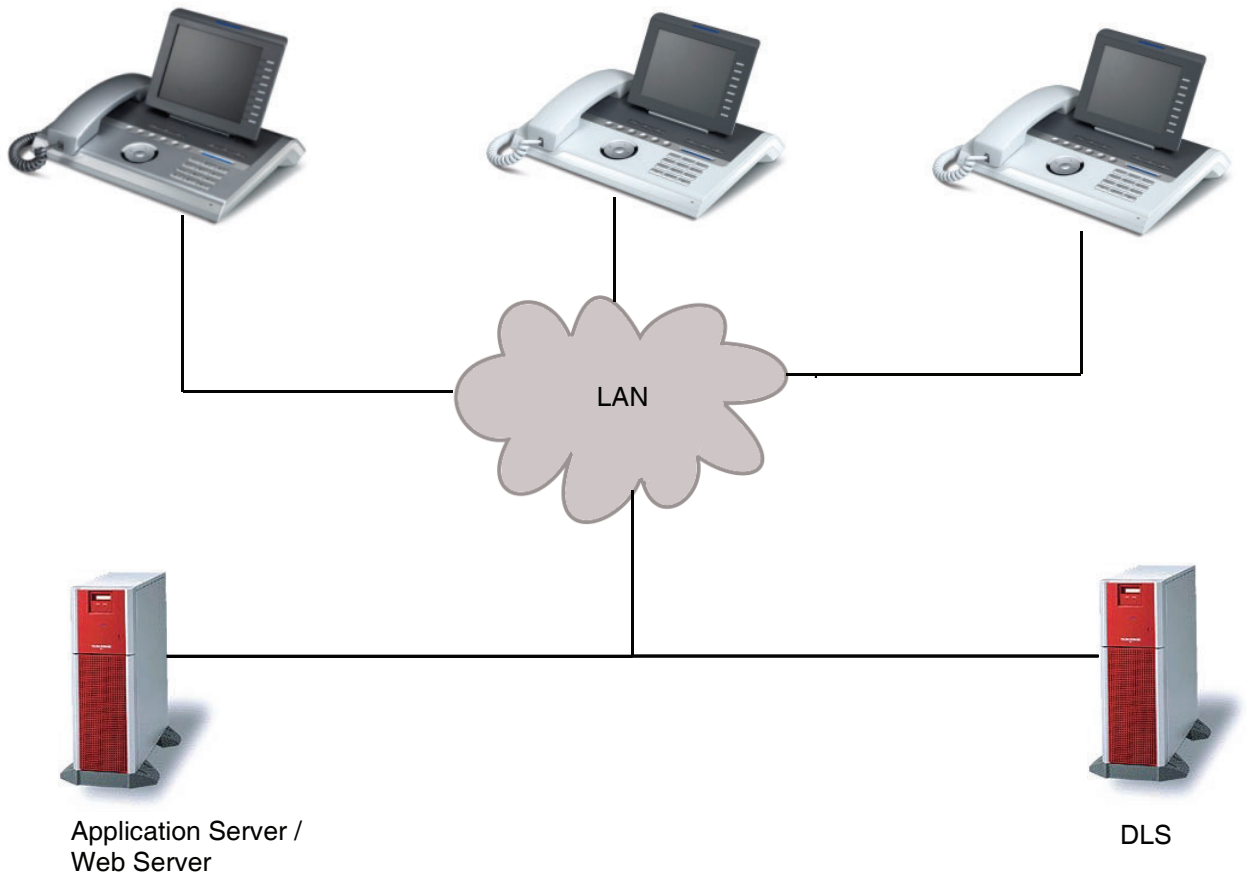
   A validating XML editor is recommended, like Oxygen, XMLWriter, EditiX, XMLSpy.
4. For developing Java Servlets: An applicable development environment, for instance Eclipse, NetBeans, IBM Websphere Studio.

2 Operating Mode and Features

2.1 Infrastructure

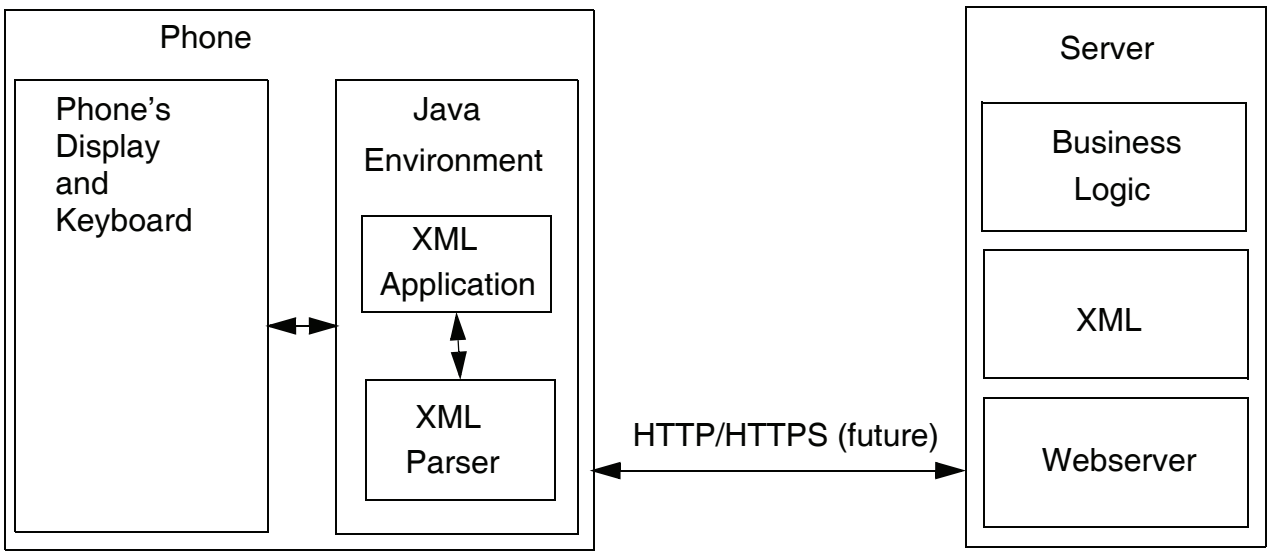
The phones are connected to a LAN. The application/web server can be connected to the same LAN, or alternatively be reachable via WAN/internet. Optionally, the DLS (Deployment Service) can be used to configure XML applications on the phone.

The following diagram shows the communication between the components:



2.2 Basic Operating Mode

The following diagram shows the basic components involved in the operation:



The operation can be described as follows:

1. The administrator configures an application on the OpenStage phone.
2. The user selects a program from the list of available applications and starts it.
3. The phone sends an HTTP request to the URL of the server-side application.

> The HTTP/HTTPS (future) GET request sent by the phone on application startup contains the phone's IP address and call number as well as the device type.
> Example:
> `137.223.238.174/serverProgram&phonenumber=4711&devicetype=OpenStage`

4. The server-side program generates an XML document which is valid according to the XML schema (see Section 6.2, "XML Schema"). The web server delivers the XML document to the phone over a HTTP connection.
5. In the phone, the XML document is parsed and displayed on the graphic display.
6. The user enters commands and data using the phone's TouchGuide and keypad.
7. Listeners in the phone software recognize the commands and data entered by the user.
8. The phone transmits the user entries to the server-side program in the form of key/value pairs.

2.3 Push Mode

The push mode enables the server-side program to control and start a phone-side XML application.

The basic operation of the Push capability can be described as follows: A program running on a server calls a special URL of the phone, which is based on the phone's IP address. Through this, a specific XML application on the phone is triggered to load an XML document from a URL given in the push request. If the specific XML application is not already running on the phone, it may be started automatically.

2.4 Cache

The cache stores up to 20 screens of any type (e.g. Form, List, Text Box, Alert, or Player) on the phone. When the cache is full, all of the entries in the cache are shifted one position to the left and the new screen is added to the end.

If the cache consists of only one full screen and the rest alerts, the shifting of the entries in the cache will retain the full screen so that there is a screen to go back to when all of the alerts have been dismissed.

Any screen in the cache may have links to other screens in the cache. They are linked by BACK and SCREEN commands (see Section 4.6, "Command"). An alert or player is removed from the cache when there are no links to it from any other screens.

When new screens are received from the remote server, the new screen may:

- Reuse the existing displayables of an existing screen in the cache,
- Overwrite an existing screen in the cache, or
- Create a new screen in the cache.

When a screen is overwritten, the existing displayables are deleted and a new set of displayables are created for the new screen.

The following table details the list reuse and screen caching rules when a new screen is received:

Screen Type	Has BACK Command		No BACK Command	
	Screen ID already exists in cache	**Screen ID does not exist in cache**	**Screen ID already exists in cache**	**Screen ID does not exist in cache**
Alert	When alerts are pushed, a new alert is created. Otherwise, the existing alert is overwritten.	A new alert is created.	When alerts are pushed, a new alert is created. Otherwise, the existing alert is overwritten.	A new alert is created.
Form	An existing form is overwritten.	A new form is created.	An existing form is overwritten.	A new form is created.
List	A new list is created.	A new list is created.	The existing list is reused.	A new list is created.
Player	The existing player is overwriten.	A new player is created.	The existing player is overwriten.	A new player is created.
Text Box	The existing text box is overwritten.	A new text box is created.	The existing text box is overwritten.	A new text box is created.

Table 2-1

3 Setup and Program Start

3.1 Configuring an Application on the Phone

For setting up an XML Application, the WBM (Web Based Manager), the local admin menu, or the DLS (Deployment Service) can be used. For the latter, please refer to the Deployment Service Administration Manual.

A maximum number of 20 XML applications can be configured on OpenStage 60/80 phones.

Please ensure that the following prerequisites are fulfilled:

- Firmware version SIP V1R3.x or higher, or HFA V2R0.x or higher
- The terminal number is set (for details, see the Administration Guide, chapter 2.3.2)
- Intranet/internet access, directly or via proxy. For details about setting a proxy, see the Administration Guide, chapter 3.13.1.2.

3.1.1 Required Data

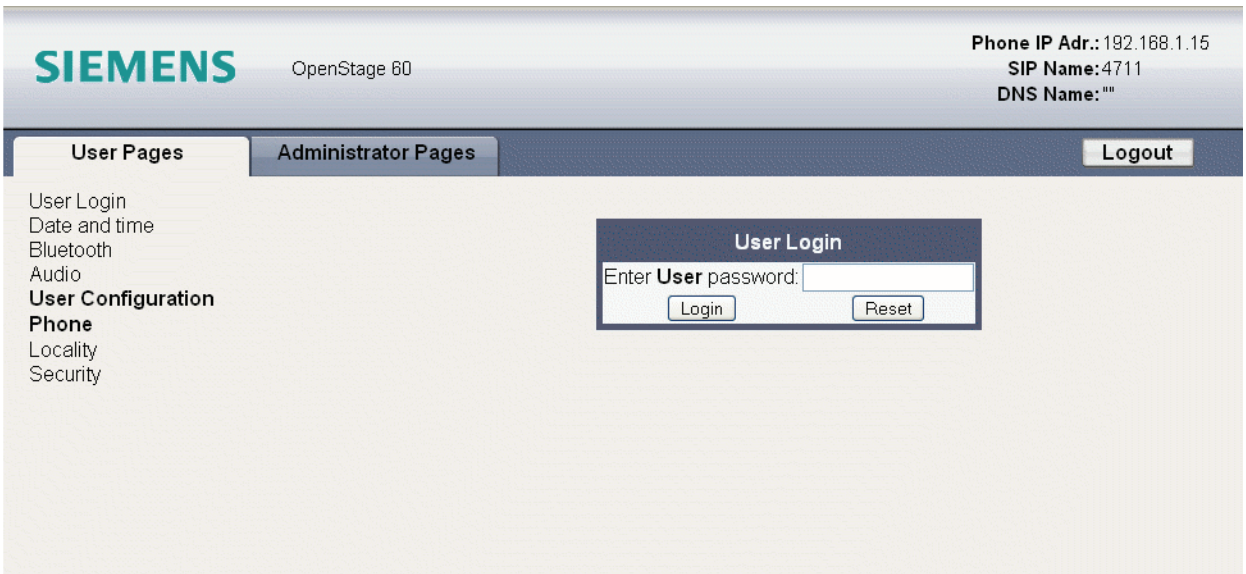
An XML application is configured by providing the following parameters:

- **Display name**: Program name to be displayed on the phone. The following criteria must be satisfied:
  - It must be unique on the phone.
  - It cannot contain the '^' character.
  - It cannot be empty.
  - Its length cannot not exceed 20 characters.
- **Application name**: This is used internally to identify the XML application running on the phone. The application name must meet the following criteria:
  - It must be unique on the phone.
  - It cannot contain non-alphanumeric characters, spaces for instance.
  - The first character must be a letter.
  - It must not be empty.
  - Its length must not exceed 20 characters.

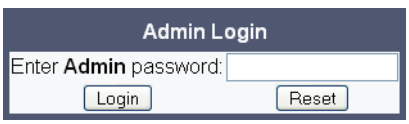
- **Protocol**: The protocol used for communication to the server: HTTP or, in the future, HTTPS.
- **Server address:** The IP address of the server that provides the application or the XML document.
  Example: `192.168.1.133`.
- **Server port number:** The number of the port used by the server to provide the application or XML document.
  Examples: `80` (Apache default port), `8080` (Tomcat default port).
- **Program name:** The relative path to the servlet or to the first XML page of the server-side program. The relative path refers to the root directory for documents on the web server. For instance, if an XML document is saved in
  `C:\Program Files\Apache Group\Apache\htdocs\ipp\ippTest.xml`
  the entry is:
  `ipp/ippTest.xml`.
  The program name cannot be longer than 100 characters.
- **Use proxy**: Enables or disables an HTTP proxy, if desired. Otherwise, a direct connection is used. For details, see Section 3.2, "Setting a Proxy".
- **XML trace enabled**: Enables or disables the debugging of the XML application back to the designated program specified by **Debug program name**. When enabled, trace info of the XML elements and details of key internal objects are sent to the remote debug program.
- **Debug program name**: The relative path to a special program on the same server as the program specified by the **Program name**. This program receives the debug information.
- **Number of tabs**: Required if the application has internal tabs. Specifies the number of internal tabs for this application. The maximum value is 3.
- **Tab 1...3 Display Name**: Required if the application has internal tabs. Provides the label text for the specified tab.
- **Tab 1...3 Application Name**: Required if the application has internal tabs. Provides a unique name for the specified tab. The remote program will use this name to provide the tab with specific content. For more information, please refer to Section 3.4, "Application Suite Using Predefined Tabs" and Section 3.4, "Application Suite Using Tabs On Demand".
- **Restart after change**: If checked, a running XML application is automatically restarted after it has been modified. After the XML application has restarted, this option is automatically unchecked.

3.1.2 Using the Web Interface

1. Open your web browser (MS Internet Explorer or Firefox) and enter the transfer protocol, IP address and port number of your phone. If HTTP is used, port 8085 must be added, for example `http://192.168.1.15:8085`. For HTTPS, the phone uses the standard port `443`.
   If the browser displays a certificate notification, confirm it. The start page of the web interface appears. In the upper right corner, the phone number, the phone's IP address, and the DNS name assigned to the phone are displayed. The left column contains the user menu tree.

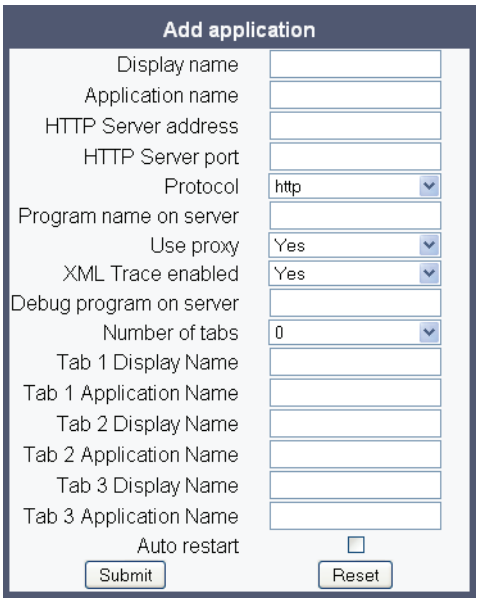


2. Click on the **Administrator Pages** tab. In the dialog box, enter the administrator password:



3. The administration main page opens. The navigation column to the left contains the menu tree. If you click on an item which is printed in normal style, the corresponding dialog opens in the center of the page. If you click on an item printed in bold letters, a sub-menu opens in the navigation column.

4. Navigate to **Applications** > **XML applications** > **Add application** and enter the required data (see Section 3.1.1, "Required Data"). When the configuration data is correct and complete, click on **Submit**.



5. Now you can test the application. Press the ☰ key on the phone to activate the **Applications** tab. If everything went well, the new application is listed and can be started by pressing OK.

3.1.3 Using the Local Admin Menu

The basics for accessing the phone's local admin menu are described here. For more detailed information, please consult the user manual or the administration manual.

1. Press the ☰ key once or repeatedly until the **Settings** tab is active. (The ☰ key toggles between the Settings menu, the Applications menu, and the applications currently running.)
2. With the TouchGuide, navigate to the **Admin** menu and press OK.
3. You are prompted to enter the administrator password.
   If the password is numeric, just type it using the number keys. On typing the first digit, an input text field will open up.
   For entering non-numeric characters, press the # key once or repeatedly, depending on the desired character. The # key cycles around the input modes as follows:
   (Abc) -> (abc) -> (ABC) -> (123) -> back to start.
   To correct an entry, use the ⮌ key.
4. Press OK. The admin menu opens.

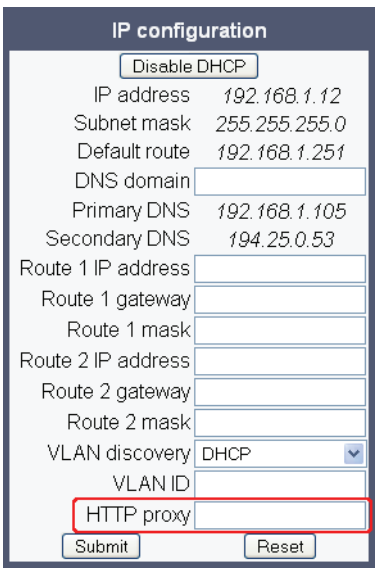
5. With the TouchGuide, navigate to the **Applications** item. To navigate, you can either run your finger around the sensor ring, or press the ▲ or ▼ key.
6. Press OK to open the Applications item and select **XML**. On OK, **Add application** appears in the context menu.
7. Press OK to open the menu for configuring new application.
8. In the application menu, enter the configuration data (for details, see Section 3.1.1, "Required Data"):
   - **Display name**: Application name for display.
   - **Application name**: Internal name for the application.
   - **Protocol**: Client/server communication protocol (HTTP or, in the future, HTTPS).
   - **Server address**: IP address of server hosting the program.
   - **Server port number**: Server port used by the server-side program.
   - **Program name**: Path to the server-side program.
   - **Use proxy**: Use of a direct connection or proxy for client/server communication.
   - **XML trace enabled** (optional): Sending trace data to a debugging program.
   - **Debug program name** (optional): Path to a server-side debugging program.
   - **Number of tabs** (optional): number of internal tabs for this application.
   - **Tab 1...3 Display Name** (optional): Label text for the specified tab.
   - **Tab 1...3 Application Name** (optional): Unique name for the specified tab.
   - **Restart after change** (optional): If checked, a running XML application is restarted automatically after it has been modified.
9. When the configuration data is correct and complete, navigate to **Save & exit** and press OK.
10. Press the ☰ key to activate the **Applications** tab. If everything went well, the new application is listed here. It can be started by pressing OK.

3.2 Setting a Proxy

The HTTP data transfer between the phone and the server on which the remote program is running can be handled by an HTTP proxy, if desired. The use of a proxy can be enabled or disabled separately for each application (see Section 3.1.1, "Required Data").

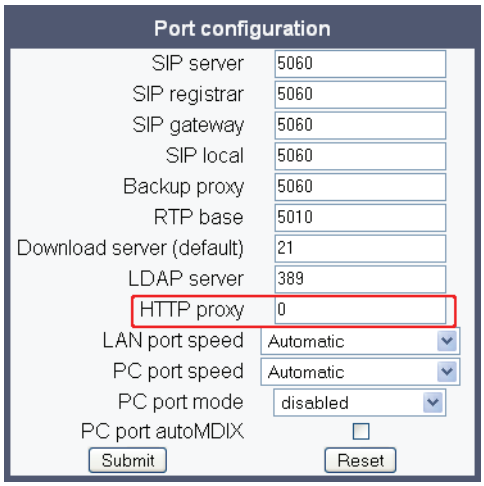
3.2.1 Using the Web Interface

1. Open the web interface as described in Section 3.1.2, "Using the Web Interface", steps 1 to 3.
2. Navigate to **Network** > **IP Configuration** and enter the IP address of the proxy to be used.



3. Press **Submit**.

4. Navigate to **Network** > **Port Configuration** and enter the port of the proxy to be used.



5. Press **Submit**. The proxy is now configured.

3.2.2 Using the Local Admin Menu

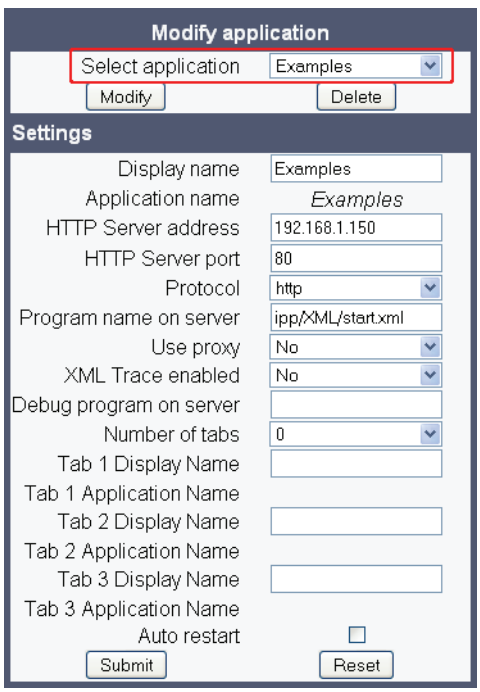
1. Repeat steps 1-4 in Section 3.1.3, "Using the Local Admin Menu".
2. With the TouchGuide, navigate to the **Network** item. To navigate, you can either run your finger around the sensor ring, or press the ▲ or ▼ key.
3. Press OK and navigate to the **IP configuration** item. On OK, the submenu opens.
4. Navigate to **HTTP proxy** and press OK.
5. In the phone's editor, enter the IP address of the proxy. For details about using the editor, please refer to the Administration Manual, chapter 3.1, step 5.
6. When the configuration data is correct and complete, navigate to **Save & exit** and press OK.

3.3 Reconfigure and Remove an Application

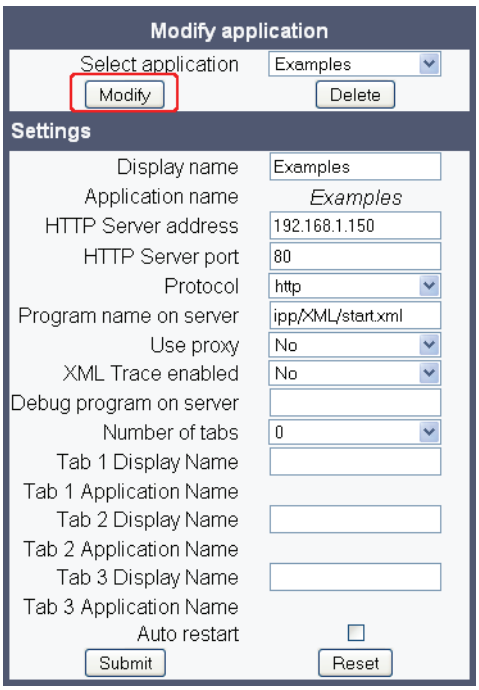
3.3.1 Using the Web Interface

3.3.1.1 Reconfigure

1. Open the web interface as described in Section 3.1.2, "Using the Web Interface", steps 1 to 3.
2. Navigate to **Applications** > **XML applications** > **Modify application** and select the application to be modified.



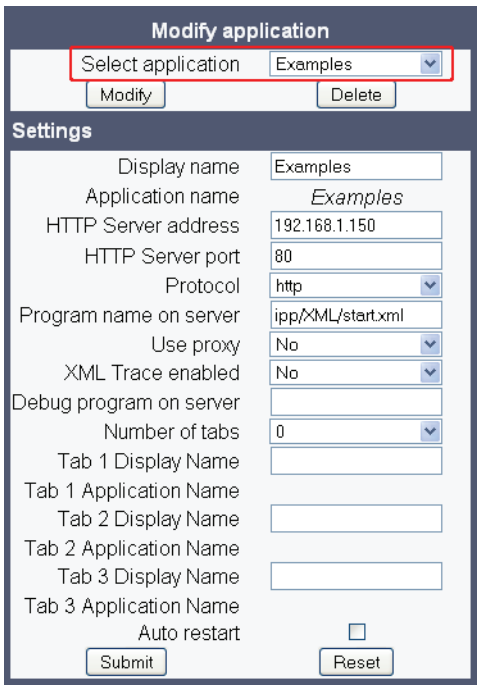
3. Click on **Modify**. The configuration data of the selected application is loaded into the input fields.



4. Change the data as desired. When the configuration data is correct and complete, click on **Submit**.

3.3.1.2 Removing

1. Open the web interface as described in Section 3.1.2, "Using the Web Interface", steps 1 to 3.
2. Navigate to **Applications** > **XML applications** > **Modify application**.
3. Navigate to **Applications** > **XML applications** > **Modify application** and select the application to be deleted.



4. Press **Delete** to remove the application.

3.3.2 Using the Local Admin Menu

The basics for accessing the phone's local admin menu are described here. For more detailed information, please consult the user manual or the administration manual.

3.3.2.1 Reconfigure

1. Repeat steps 1-5 in Section 3.1.3, "Using the Local Admin Menu".
2. Press OK to open the **Applications** item and select **XML**. On OK, **Add application** and the names of the previously configured applications appear in the context menu.
3. Navigate to the program that you would like to reconfigure and press OK.
4. In the application-specific menu, enter the data required for the configuration of a program (see Section 3.1.1, "Required Data").
5. When the configuration data is correct and complete, navigate to **Save & exit** and press OK.

3.3.2.2 Removing

1. Repeat steps 1-5 in Section 3.1.3, "Using the Local Admin Menu".
2. Press OK to open the **Applications** item and select **XML**. On OK, **Add application** and the names of the previously configured applications appear in the context menu.
3. Navigate to the program that you would like to delete and press OK.
4. The **Save & exit** option is highlighted. Open the context menu by pressing the → key.
5. With the TouchGuide, navigate to **Remove & exit** and press OK.

3.4 Starting the Application

An XML application can be started by navigating to the applications menu using the ☰ key, by configuring a programmable key for a specific application, or, in case of an Xpressions or phonebook application, by the corresponding mode key. For details, please refer to the OpenStage User Manual and the OpenStage Administration Manual.

The HTTP GET request sent by the phone depends on the target URL. If this URL has an `.xml` extension, the phone assumes to receive a static XML document, e. g. from a web server. The request URI will contain no additional information. If the requested URL has no `.xml` extension, the phone assumes it is communicating with a dynamic application, e. g. a servlet. In this case, the phone will include its IP address, telephone number, and device type using key/value pairs in the request. The request has the following format:

```
http://<server ip address>:<port>/<program name>?ipaddress=<phone
IP>&phonenumber=<phone number>&devicetype=OpenStage
```

In both cases, whether the request URL has an `.xml` extension or not, the current language is sent to the server in the `Accept-Language` field of the HTTP request header, for example:

```
Accept-Language: en
```

Application Suite Using Predefined Tabs

In addition to single applications, it is possible to run an application suite, which is divided in up to 3 internal tabs. With regular XML applications and phonebook applications, these tabs can be predefined by the administrator. For configuring an application suite, please refer to Section 3.1, "Configuring an Application on the Phone".

The HTTP GET requests sent by the phone depend on the target URL. If this URL has an `.xml` extension, the phone will request the same XML document for each tab. If it has no `.xml` extension, the phone will append the unique tab name to the request URI. The request has the following format:

```
http://<server ip address>:<port>/<program name>?ipaddress=<phone
IP>&phonenumber=<phone number>&devicetype=OpenStage&tab=<unique tab
name>
```

Depending on how many tabs have been configured for the application, the phone will send 1 or 2 further similar requests for the other tabs. Each of these requests will be sent right after the response to the preceding request has been received. With regular XML applications, the user can switch between the tabs using the ☰ key. With Xpressions applications, the ✉ key is used for tab switching, and with phonebook applications, the 🕮 key is used.

Application Suite Using Tabs On Demand

As an alternative to predefined tabs, it is possible to start tabs on demand using Push requests (Section 2.3, "Push Mode" and Section 5.1, "The 'Push' Request") from the server. Xpressions applications always use tabs on demand. With tabs on demand, there can be different content on every tab. The first tab started uses the start URL configured on the phone. The remaining tabs use the URL supplied by the remote server.

The focus is on the first tab started by the phone. With Xpressions applications, the ✉ key is used for tab switching, and with phonebook applications, the 🕮 key is used.

3.5 Mobile Users And Privacy

As some XML applications may process personal user data, it is important that the remote application is aware of the user currently logged onto the phone. This can be achieved by mapping the phone number against the phone's IP address. Both parameters are available to the remote program, as they are sent by the phone on application start. Please note that this will not be the case if the program name has an `.xml` suffix. For details please see Section 3.4, "Starting the Application".

4 XML Object Reference

This chapter describes the usage, content and error handling for the individual XML objects. The objects are comprised of a main element, its attributes and one or more child elements and their attributes. The names of the basic XML elements are prefixed with "Ipp", which stands for "IP phone", for example, "IppAction".

For each individual XML object, an example is provided. All examples from this document are available for download under

http://wiki.siemens-enterprise.com/images/0/04/OpenStage_XML_apps_examples.zip

The following table gives an overview of the individual objects and their functions:

Category	Object	Functional Description
Basic XML structure	Phone	Optional root element; for future objects not shown on the display.
	Display	Root element for objects within the display area.
Navigation	Command (BACK, SCREEN, UP-DATE)	BACK: Leads to the screen displayed previously. SCREEN: Leads to a specific screen. UPDATE: Reloads the screen from the server.
Form	Form	Form for sending the contents of data fields to the server.
Message win-dow	Alert	Temporarily appears over the currently displayed screen and displays a message.
Ticker	Ticker	Ticker text; displayed underneath the title bar.
Text, layout, image	Form/StringItem	Shows text on the display.
	Image	Links an image file.
	Form/ImageItem	Positions an image.
	Form/Spacer	Creates a vertical space.
Selection lists	List	List of selectable options.
	Form/ChoiceGroup	List of selectable options.
Data input	TextBox	Text input field with multiple lines.
	Form/TextField	Text input field with a single line.
	Form/DateField	Date input field.
	Form/Gauge (inter-active)	Entry of a numerical value using a horizontal slider.

Table 4-1

Category	Object	Functional Description
Hidden data transfer	Hidden	Invisible field for transmitting data between telephone and server
Visualization of chronological processes	Form/Gauge (non-interactive)	Horizontal bar that grows every second
Sending data to the server	Command (SELECT)	The data contained in the entry fields on the screen are sent to the server.
	Form/Button	Sends the data that is saved in attributes to the server
Telephony	Action (MAKECALL)	Initiates a call to a specific telephone number
LED control	Action (TURNLEDON or TURNLEDOFF)	Turns on/off the LED on the Message mode key

Table 4-1

4.1 Encoding/Internationalization

The OpenStage XML interface supports UTF-8 and ASCII. If a different character set is used, e. g. ISO 8859-1, special characters can be encoded by numeric character references.

Example: `ä` will be rendered as the german umlaut "ä".

For details, see http://www.w3.org/TR/2000/REC-xml-20001006#sec-references.

Some text editors add the leading sequence `0xEF 0xBB 0xBF` to an UTF-8 file. This is a byte-order mark (BOM), which is conventionally used to indicate that the text is encoded as UTF-8. As the phone's XML parser does not support UTF-8 files with BOM, please make sure you use an editor that does not insert this mark. The problem exists, for instance, with older versions of Notepad++. There are reportedly no problems with Notepad2, Notetab, or Altova XML Spy.

4.2 Phone

`IppPhone` is an optional root element which enables the integration of special objects in the XML application. These objects are not displayed. They will be available in future versions of the OpenStage 60/80 phone software. Using `IppPhone` in current XML applications has the advantage of allowing easy extension in the future.

The use of `IppPhone` as the root element is not mandatory, `IppDisplay` (see Section 4.3, "Display") can also server as the root element of an XML application.

Syntax

```
<IppPhone>
   <IppDisplay/>
</IppPhone>
```

4.3 Display

This object enables multiple Screens within a single XML document. A Screen corresponds to the content that is simultaneously shown on the display, comparable to a web page. Accordingly, the `IppDisplay` element functions as the root element of an XML document (unless `IppPhone` is used; see Section 4.2, "Phone"). Therefore, as the container for Screen objects, it is comparable to cards in a WML document.

An XML document may contain up to 5 screens, one of which may be the "InitialScreen" and another the "UpdateScreen". If there are more than 5 screens in the XML document, the additional screens are ignored.

The following attributes are available to the `IppDisplay` element:

Attribute	Value	Functionality/Remarks
`UpdateScreen` (optional)	ID of the second screen to be updated (integer)	Used when two Screens have to be updated simultaneously. Provides the ID of the second Screen to be updated.
	Missing, negative, or invalid	There is no second screen to update.

Syntax

```
<IppPhone>
   <IppDisplay UpdateScreen="[ID]">
      <IppScreen/>
      <IppScreen/>
      <IppScreen/>
      <IppScreen/>
      <IppScreen/>
   </IppDisplay>
</IppPhone>
```

4.4 Screen

The `IppScreen` element serves as a container for specified contents. The `ID` attribute enables switching between different screens.

An XML document may contain up to 5 screens, one of which may be the "InitialScreen" and another the "UpdateScreen". If there are more than 5 screens in the XML document, the additional screens are ignored.

If a new Screen is loaded whose Screen ID already exists, elements of cached Screens may be overwritten. For details, see the table in Section 2.4, "Cache".

The following attributes are available to the `IppScreen` element:

Attribute	Value	Functionality/Remarks
`ID` (optional)	ID of the screen element (integer)	Enables referencing of the Screen object, so that the Screen can be accessed directly. If missing, negative, or invalid, it will default to 0.
`HiddenCount` (mandatory)	Number of Hidden objects	All Hidden objects are displayed.
	`0`	All Hidden objects are ignored. If missing, negative, or invalid, it will default to 0.
	A number lesser than the actual number of Hidden objects	All the Hidden objects that exceed the number indicated are ignored.
`CommandCount` (mandatory)	Number of Command objects	All Command objects are displayed. If missing, negative, or invalid, it will default to 0.
	`0`	All Command elements are ignored.
	A number lesser than the actual number of Command objects	All the Command elements that exceed the number indicated are ignored.

Syntax

The IppScreen element must contain one of the following elements:

- `IppAlert`
- `IppList`
- `IppTextBox`
- `IppForm`

```
<IppPhone>
   <IppScreen ID="[ID]" HiddenCount="[Number]" CommandCount=
"[Number]">
      <IppKey/>
      <IppAction/>
      <IppCommand/>
      <IppAlert/>
      <IppList/>
      <IppTextBox/>
      <IppTicker/>
      <IppHidden/>
      <IppForm/>
      <IppPlayer/>
   </IppScreen>
</IppPhone>
```

4.5 Action

Actions provide a request mechanism to the phone to perform specified tasks. For example, make a call to a specified number and end the call.

For security reasons, an XML application can only clear a call that it has initiated. Thus, in order to be able to clear a call down, the application must be running whilst the call is in progress. If the application is stopped and restarted whilst the call is in progress, the newly started application can not clear the call, because it did not initiate it.

If Xpressions is configured on the phone, the LED on the message mode key can be switched on or off by a special action type.

Attribute	Value	Functionality/Remarks
TYPE (mandatory)	MAKECALL	A call to the telephone number specified in the child element <Number> is initiated. The possible formats for the call number depend on the phone's dial settings. Please refer to the Administration Manual of the phone, chap. 3.8.
	ENDCALL	Ends the call made by a MAKECALL action.
	TURNLEDON	Turns on the LED on the message mode key. (Only available if Xpressions is configured.)
	TURNLEDOFF	Turns off the LED on the message mode key. (Only available if Xpressions is configured.)
	Missing or invalid	No action will be performed.

Syntax

```
<IppAction Type="MAKECALL|ENDCALL|TURNLEDON|TURNLEDOFF">
   <Number>[Phone Number]</Number>
</IppAction>
```

Child elements

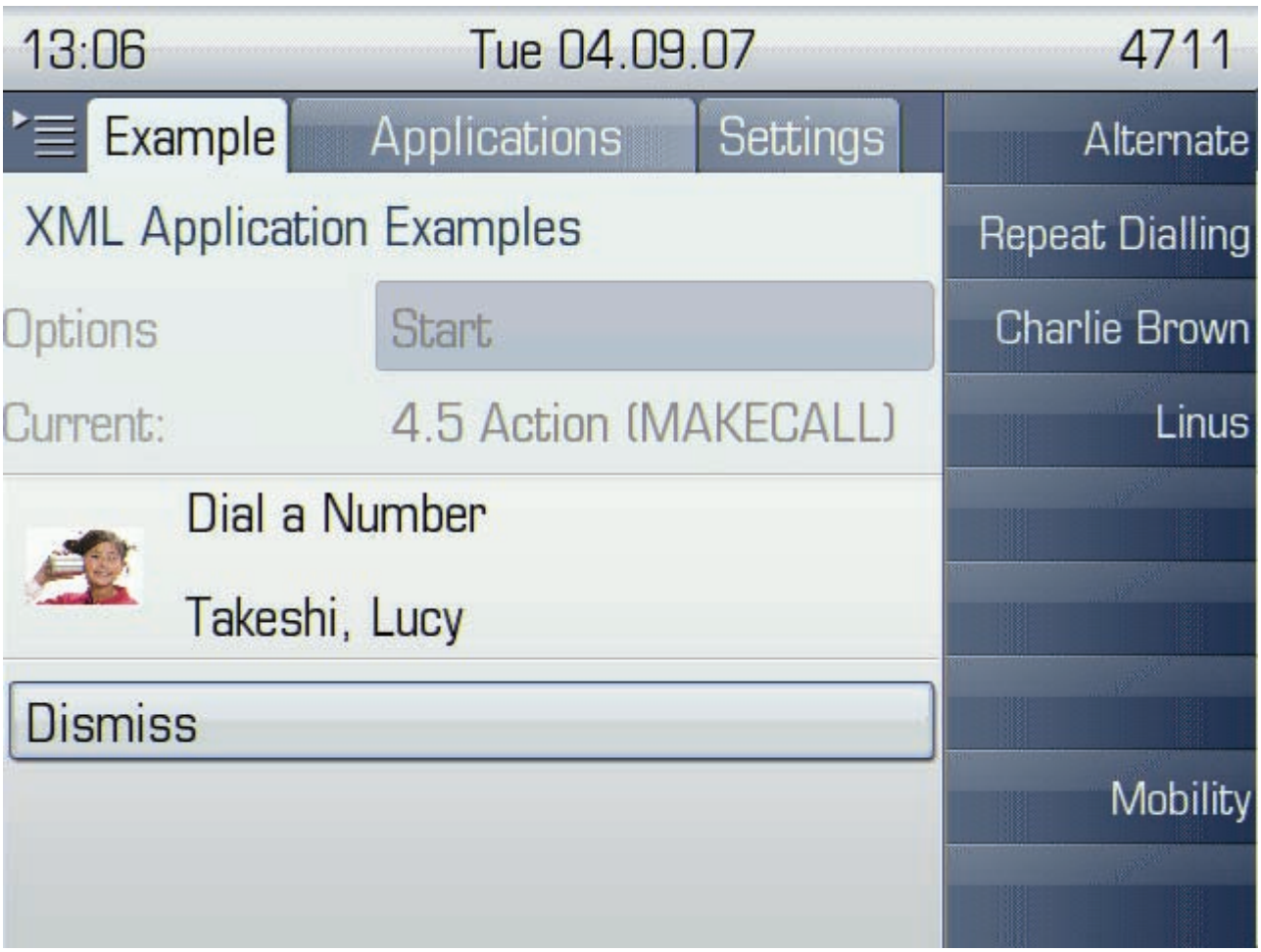
- Number

  The telephone number to be dialed.

Example

```
<?xml version="1.0" encoding="UTF-8" ?>
<IppPhone>
   <IppDisplay>
      <IppScreen ID="1" HiddenCount="0" CommandCount="0">
         <IppAlert Type="CONFIRMATION" Delay="FOREVER">
            <Title>Dial a Number</Title>
            <Text>
              3290
               <IppPhoneNumber ImageType="PICTURECLIP"
NumberType="NAME">
                   <AltText>No image available for this entry</AltText>
               </IppPhoneNumber>
            </Text>
            <Image></Image>
         </IppAlert>
         <IppAction Type="MAKECALL">
            <Number>+49897223290</Number>
         </IppAction>
      </IppScreen>
   </IppDisplay>
</IppPhone>
```

Screenshot


13:06
Tue 04.09.07
4711
Example
Applications
Settings
Alternate
XML Application Examples
Repeat Dialling
Options
Start
Charlie Brown
Current:
4.5 Action (MAKECALL)
Linus
Dial a Number
Takeshi, Lucy
Dismiss
Mobility

4.6 Command

Commands generally appear in context menus which are associated either with the Options menu or with the clickable area of a specific element. However, CANCEL and EXIT commands do not appear in the Options menu. Some Commands are associated with TouchGuide keys on the phone. With Alerts, the commands are displayed at the bottom of the alert.

The following attributes are available to the `IppCommand` element:

Attribute	Value	Functionality/Remarks
`Type` (mandatory)	`SELECT`	This transmits the key/value pairs from the current List, Text Box, Form, or Player, to the URL indicated there. Key/value pairs from Hidden elements and the key/value pair associated with the Command, if present, are also sent to the server.
	`BACK`	Switches to the cached version of the Screen displayed previously. The command will always go back to the previous full screen, i. e. Form, List, or Text Box, but not Alert or Player. If there is only a single screen in the XML document, no menu item is generated.
	`UPDATE`	Replaces the current Screen with the Screen specified in the current List, Text Box, Form, or Player. The data for this Screen is fetched from the server using the URL associated with the current screen. Key/value pairs from Hidden elements and the key/value pair associated with the Command, if present, are sent to the server. As opposed to a SELECT Command, no key/value pairs from the current List, TextBox, Form, or Player, are sent to the server.
	`SCREEN`	Switches to the Screen specified in the child element `ScreenID`. The server-side program is not informed of this action. If the data in `ScreenID` are missing or invalid, the current screen will continue to be displayed.
	`CANCEL`	Used to undo a selection made in selection lists (List, Form/ChoiceGroup) or an entry made in input fields (TextBox, Form/TextField, Form/DateField). The contents of the current screen revert back to their default values cached on the phone.
	`EXIT`	Ends the program.

`Priority` (optional)	Priority (integer)	This determines the order of the Command objects, whereby `0` represents the one with the highest priority, followed by `1` and so on.
`Auto` (optional)	Total duration in seconds	The Command object is automatically executed after the specified time, that is, without the user doing anything. If multiple Command objects containing the Auto attribute exist, the one that is set for the shortest total duration is executed. This means that only one command per Screen is executed automatically.
	`0` or missing, or negative, or invalid	The Command is only invoked through user action.
`Key` (optional)	Key	The key for the key/value pair that is sent to the server as a HTTP GET request. Please note that the Key must be unique within a Screen.
`Value` (optional)	Value	The value for the key/value pair that is sent to the server as a HTTP GET request.
`DisplayOn` (optional)	`OPTIONS`	Default value. The Command is displayed from the "Options" bar.
	`LISTITEM`	The Command is displayed on a List item (see Section 4.9, "List") or on an individual item in a Form, if included in a Form Item (see Section 4.13.8, "Form/Item").
	`BOTH`	The Command is displayed both on the "Options" bar and on List items or Form items.
`Select` (optional)	`YES`	The Command is set as the select command for an Implicit List (see Section 4.9, "List"). If more than one Command is set as the `Select` command for an Implicit List, only the first one will be used as the selection action. Pressing OK when the "Options" bar is highlighted returns the key-value pair for the Select command. Pressing OK when an individual list item is highlighted returns the key-value pairs for both the Select command and the highlighted list item.
	`NO`	Default value. The Command is not set as the select command for an Implicit List

`Default` (optional)	`YES`	The Command is set as the default command in the context menu. If more that one Command is set as the default command, only the first one will be the default command.
	`NO`	Default value. The Command is not set as the default command in the context menu.

Syntax

```
<IppCommand Type="SELECT|BACK|UPDATE|SCREEN|CANCEL|EXIT"
Priority="[Number]" Auto="[Number]" Key="[Key]" Value="[Value]"
DisplayOn="OPTIONS|LISTITEM|BOTH" Select="YES|NO" Default="YES|NO">
   <Label> </Label>
   <ScreenID> </ScreenID>
</IppCommand>
```

Multiple occurrences of a type can appear on a screen. In such cases, the commands should be distinguishable by the user.

If the `Type` attribute is missing or has an invalid value, it is not displayed. If the child element `Label` is missing, a button is displayed with the Command type instead of a label, e. g. "SELECT".

Child elements

- `Label`

  Contains the label for the menu item that is generated for the Command.

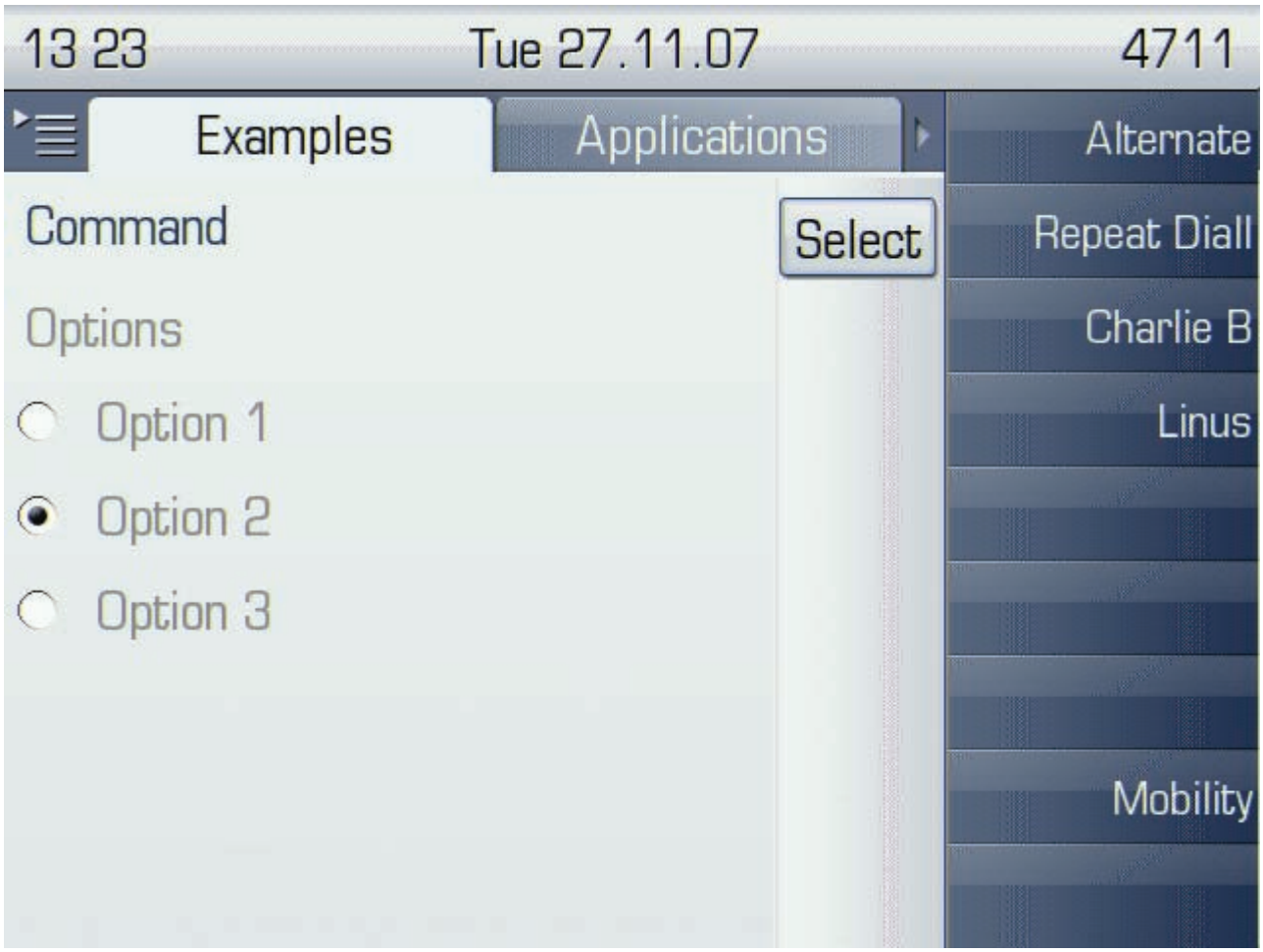
- `ScreenID`

  Indicates which screen should be accessed next on activating a `SCREEN` Command. The `ScreenID` is only appropriate if the Command type is `SCREEN`. If the `ScreenID` is missing or invalid, the current Screen will not be replaced, but continue to be displayed.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<IppPhone>
   <IppDisplay>
      <IppScreen ID="1" HiddenCount="0" CommandCount="2">
         <IppList Type="EXCLUSIVE" Count="3">
            <Title>Command</Title>
             <Url>http://subdomain.domain/path/program</Url>
             <Option ID="1" Selected="FALSE" Key="key" Value="1">
                <OptionText>Option 1</OptionText>
             </Option>
             <Option ID="2" Selected="TRUE" Key="key" Value="2">
                <OptionText>Option 2</OptionText>
             </Option>
             <Option ID="3" Selected="FALSE" Key="key" Value="3">
                <OptionText>Option 3</OptionText>
             </Option>
         </IppList>
         <IppCommand Type="SELECT" DisplayOn="BOTH" Priority="1">
            <Label>Select</Label>
            <ScreenID>2</ScreenID>
         </IppCommand>
      </IppScreen>
   </IppDisplay>
</IppPhone>
```

Screenshot


13 23
Tue 27.11.07
4711
Examples
Applications
Alternate
Command
Select
Repeat Diall
Options
Charlie B
Option 1
Linus
Option 2
Option 3
Mobility

4.7 Direct Keypad Input

4.7.1 Overview

In addition to regular user input by the means of input fields, it is possible to send keypad entries to the remote server in realtime, or, alternatively, in buffered mode. In unbuffered mode, the phone will immediately send an HTTP GET request to the specified server when a keypad key has been pressed. After this, the phone will wait for a server response. The request contains a key/value pair wherein the key is predefined in the XML document, and the value represents the value of the keypad key.

The phone itself will not generate audible or visual feedback to the user when the data is sent; this must be handled by the remote server.

The feature is available for lists (see Section 4.9, "List"); with alerts, forms, text boxes, or players, only regular user input is supported. To enable and configure direct key input, the `IppKey` element is used.

To add context information, `IppHidden` (Section 4.12, "Hidden") elements can be used. The key/value pairs contained in the `IppHidden` elements will be sent along with each keypad input. For example, the remote server may keep track of the entire digit sequence entered so far by including this digit sequence in an `IppHidden` element.

Long presses of the * and # keypad keys toggle the ringing state and invoke phone lock. As the keypad input feature does not distinguish between short and long press, these functions will occur in parallel with sending keypad input to the remote server. If the remote server updates an XML application whilst the phone is locked, the user does not see the updated screen until after the phone is unlocked.

Certain keypad key combinations may be invoked on the phone to enter the admin menu, invoke a factory reset, or restart the phone. When the first keypad key of a combination is pressed, the corresponding value will be sent to the remote server. However, the subsequent key presses of the key combination will not be communicated.

If an alert (see Section 4.8, "Alert") is displayed over a list which had sending keypad input enabled, sending keypad input will be automatically disabled. Moreover, it will stay disabled when the alert is dismissed and the list has focus. To re-enable sending keypad input on the list, the XML document for the list must be resent.

Buffering

As an alternative to immediate sending and interaction with the server, keypad input can be buffered and sent afterwards. This is controlled by the `BufferKeys` element. Depending on the settings of `BufferLength` and `TermKey`, the data is submitted either when the key sequence has reached the predefined buffer length, or when the predefined termination key is pressed.

If an application requires each keypad key press one at a time, the remote server should request subsequent buffering (see Section 4.7.3.4, "Buffer Subsequent Keypad Key presses"). This ensures that no keypad key presses are thrown away when the user types keypad keys quicker than responses are received from the server.

Error Handling

If there is a failure in the communication protocol between the phone and the remote server when sending keypad key presses to the remote server, no error message is displayed to the user. However, if requested by the server via XML document, the key presses will be buffered on the phone. When the user presses the next keypad key, and the connection is re-established, that key press and all buffered key presses will be sent to the remote server.

4.7.2 Possible Uses

Potential example XML applications using sending keypad input are:

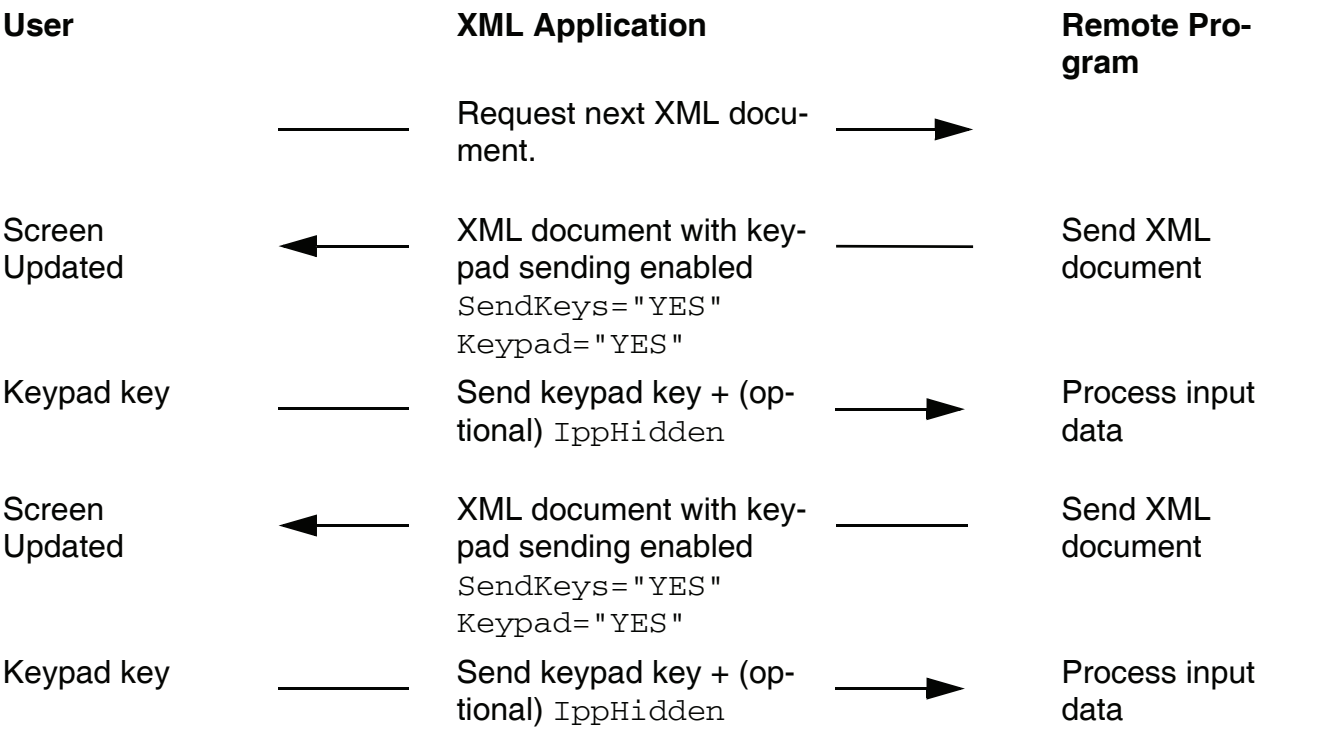
1. A phonebook application in which the user initiates a search for the appropriate entry using the keypad keys. The digits are sent to the remote program. The remote program uses the digits as search criteria and returns a list of matching entries. These entries are immediately displayed to the user. If the user presses subsequent keypad keys after the first keypad digit, and a response has not yet been received, the keypad digits are buffered, and sent en bloc to the remote program.
2. A menu application provides a sequence of numbered lists of options. The user navigates in a list using the keypad keys. The remote program requires single keypad digits to determine the next list. Therefore, multiple keypad digits or any subsequent keypad digits are ignored whilst the phone is waiting for server response.
3. An application requests the user to enter a PIN. The user presses a sequence of keypad digits terminated by #. The digits are buffered until the # is detected. After this, all of the digits including the # are sent en bloc to the remote program.

4.7.3 Typical Client-Server Interactions

This section describes typical interactions between the phone-side XML application and the server-side program with different keypad sending settings.

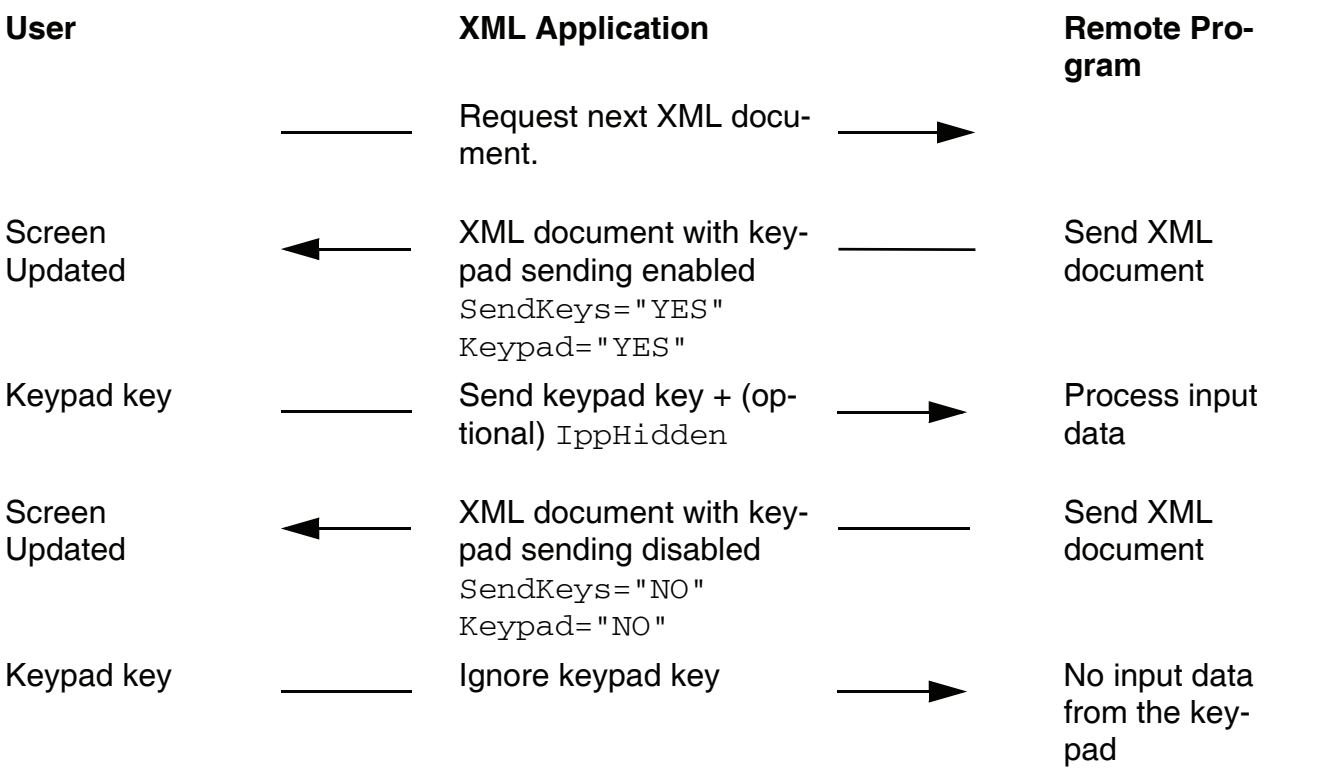
4.7.3.1 Sending Keypad Keys Enabled and Immediate Submission

In the following interaction, the remote program enables send keypad input (and disables buffering). As soon as the user presses a keypad key, the screen updates.



4.7.3.2 Disable Sending Keypad Keys

In the following interaction, the remote XML application initially enables send keypad input. The user presses a keypad key which is sent to the remote program. With the following XML document, the XML application disables send keypad input. All subsequent keypad key presses are ignored.



4.7.3.3 Buffer Initial Keypad Key Presses

In the following interaction, the remote program enables send keypad input and buffering with a buffer length greater than 1. When the user presses one or more keypad keys, the key presses are buffered until the buffer is full, i.e. the number of key presses equals the predefined buffer length. After this, the key data are sent en bloc to the remote XML application. Alternatively, a terminating keypad key is defined and the non-terminating keypad key presses are buffered until the terminating key is pressed. After this, the key data are sent en bloc, including the terminating keypad key.

User		**XML Application**		**Remote Program**
	———	Request next XML document.	→	

Screen Updated	←	XML document with keypad sending enabled, buffer keypad input enabled
		`SendKeys="YES"`
		`Keypad="YES"`
		`BufferKeys="YES"`
		`BufferLength="3"` or
		`TermKey="#"`
			———	Send XML document

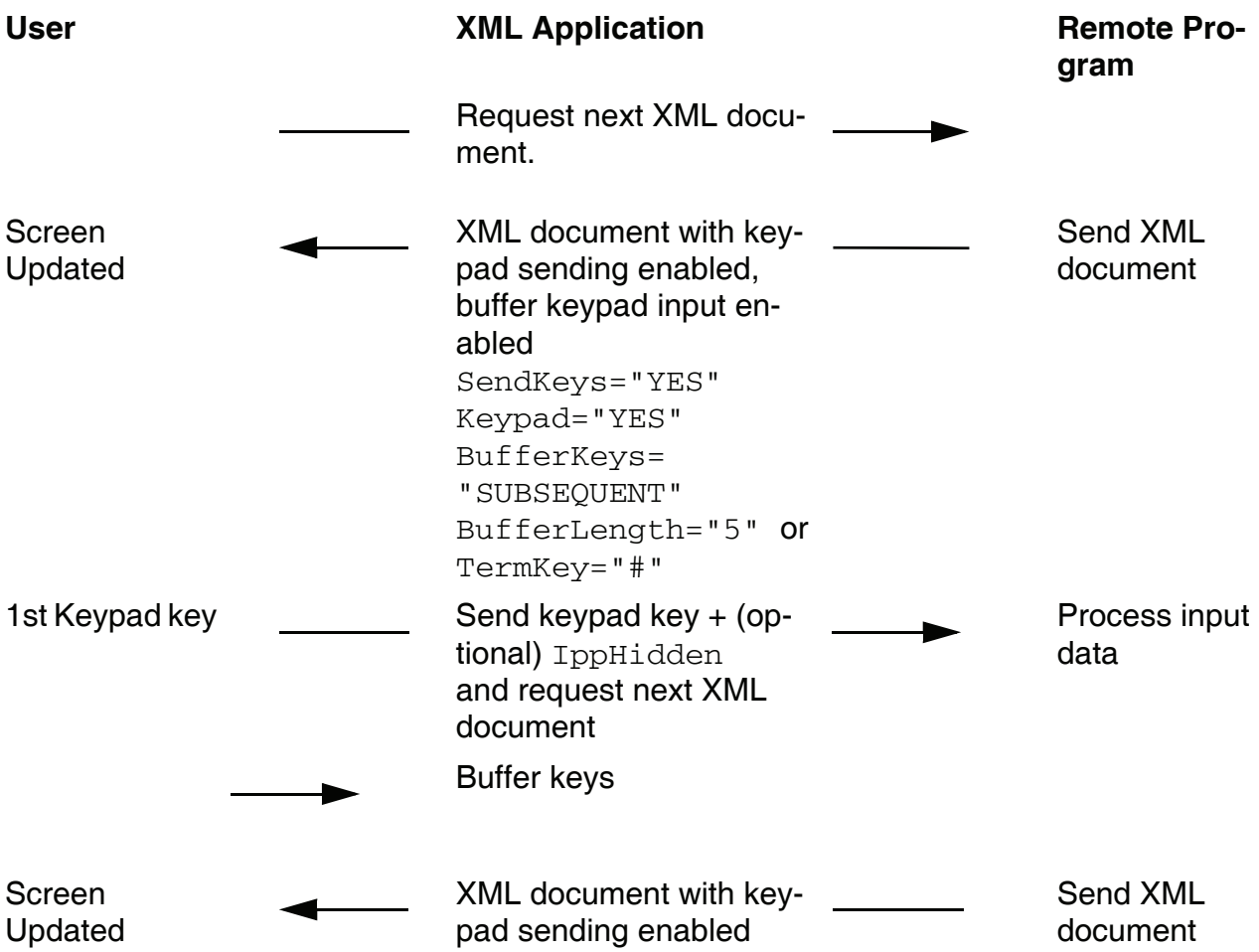
Sequence of keypad key-presses: 1st key 2nd key 3rd key or #	———	On terminating key or buffer full: Send keypad keys + (optional) `IppHidden`	→	Process input data
Screen Updated	←	XML document	———	Send XML document

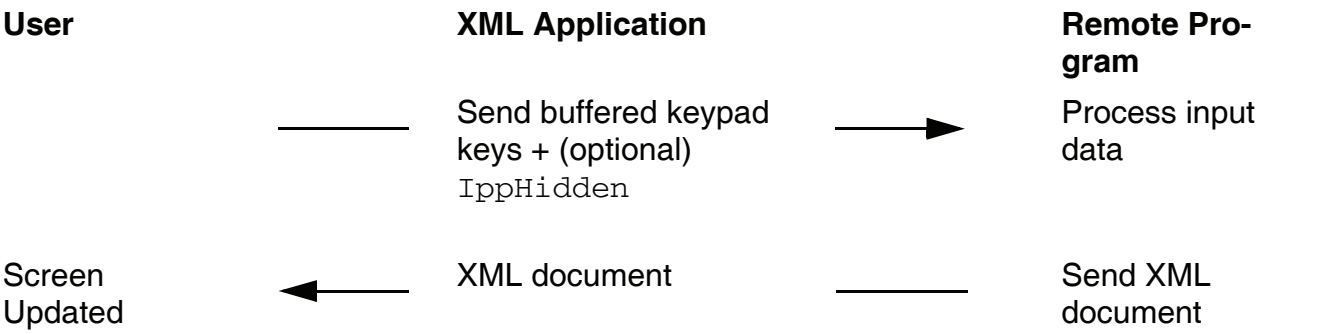
4.7.3.4 Buffer Subsequent Keypad Key presses

In the following interaction, the remote program enables send keypad input and buffer subsequent keypad keys. As soon as the user presses the first keypad key, e. g. to enter the first letter of a name to be searched in a phonebook, the key is sent to the server. Subsequently, the user types in some more letters or digits. These characters will be buffered until:

- the buffer length is reached, or
- the terminating key, if defined, has been pressed.

As soon as one of these criteria is met, and the response to the previous request issued with sending the first key has been received, the buffered keys are sent to the server. In the example interaction, the response from the server is received first, that is, before the buffer length is reached or the user has typed in the terminating key.


User
XML Application
Remote Program
Request next XML document.
Screen Updated
XML document with keypad sending enabled, buffer keypad input enabled SendKeys="YES" Keypad="YES" BufferKeys= "SUBSEQUENT" BufferLength="5" or TermKey="#"
Send XML document
1st Keypad key
Send keypad key + (optional) IppHidden and request next XML document
Process input data
Buffer keys
Screen Updated
XML document with keypad sending enabled
Send XML document


User
XML Application
Remote Program
Send buffered keypad keys + (optional) IppHidden
Process input data
Screen Updated
XML document
Send XML document

The following attributes are available to the `IppKey` element:

Attribute	Value	Functionality/Remarks
`Keypad` (mandatory)	`YES`	Keypad key presses are handled.
	`NO`	Default value. Keypad key presses are ignored.
`SendKeys` (mandatory)	`YES`	Key digits are sent.
	`NO`	Default value. Key digits are not sent.
`BufferKeys` (optional)	`YES`	The key presses are buffered.
	`NO`	Default value. The key presses are not buffered, but immediately sent to the server.
	`SUBSEQUENT`	The key presses following the first key press are buffered.
`BufferLength` (optional)	Number of key presses to be buffered Default: `0`	Defines the length of the buffer. When the buffer is full, the digits are sent en bloc.
`TermKey` (optional)	`0` to `9`, `*`, `#`, or left blank Default: blank	Defines the terminating keypad key, which triggers the submission of the buffered digits. Please note that `*` and `#` are URL encoded when sent to the server.
`UrlKey` (mandatory)	User defined string Default: `digit`	Defines the key part of the key-value pair sent in the URL string to the remote server for the next XML document.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
   <IppDisplay InitialScreen="1">
      <IppScreen ID="1" HiddenCount="1" CommandCount="1">
         <IppKey Keypad="YES" SendKeys="YES" BufferKeys="YES" Buffer-
Length="3" UrlKey="mydigit"/>
        <IppList Type="IMPLICIT" Count="3">
           <Title>IppList</Title>
           <Url>http://subdomain.domain/path/program</Url>
           <Option ID="1" Selected="FALSE" Key="1" Value="1">
              <OptionText>Item 1</OptionText>
              <Image/>
           </Option>
           <Option ID="2" Selected="TRUE" Key="2" Value="2">
              <OptionText>Item 2</OptionText>
              <Image/>
           </Option>
           <Option ID="3" Selected="FALSE" Key="3" Value="3">
              <OptionText>Item 3</OptionText>
              <Image/>
           </Option>
        </IppList>
        <IppCommand Type="SELECT" Priority="0">
           <Label>Main Screen</Label>
           <ScreenID>2</ScreenID>
        </IppCommand>
        <IppHidden Type="VALUE" Key="xmlobject">
           <Value>_show</Value>
        </IppHidden>
   </IppScreen>
</IppDisplay>
```

4.8 Alert

An Alert is a message window displayed on the bottom half of the current screen. Different types of Alert can be selected using the `Type` attribute. Title, text, and an image can be specified in the child elements.

When an Alert is created, it implicitly has a special command present on it to close the Alert Screen. This command has the label "Dismiss". If the Screen containing the Alert also includes a Command, the "Dismiss" command is replaced by the Command added to the Screen. Moreover, the Alert is automatically turned into a modal Alert that can only be cleared by the user. Hence, its timeout value, which controls the persistence on the display, is always `FOREVER`.

The optional element `Url` enables explicitly added `SELECT`, `UPDATE`, and `EXIT` Commands (see Section 4.6, "Command") to be returned to the application running on the server. If the explicitly added Commands are only `BACK`, `SCREEN`, or `CANCEL`, the `Url` element is not required .

Another optional element, `IppPhoneNumber`, provides a connection to the local phone book. If the `IppPhoneNumber` element contains a phone number that exists in the phone's local phone book, the name and number of the participant are displayed. Furthermore, if available, a picture clip or, else, an icon for the phone type in question (e.g. mobile number, home number, etc.) are shown. A picture clip from the phonebook will override an image object. The translation of phone number to name, number, and picture/icon is done on the phone. If the translation fails, alternative text may be displayed instead of the phone number. The alternative text is supplied in the XML document. The `IppPhoneNumber` tag is defined in Section 4.16, "Phone Number".

The following attributes are available to the `IppAlert` element:

Attribute	Value	Functionality/Remarks
`Type` (optional)	`ALARM`	An alarm icon is displayed at the left side.
	`CONFIRMATION`	A prompt for confirmation icon is displayed at the left side.
	`ERROR`	Default value. An error icon is displayed at the left side.
	`INFO`	An info icon is displayed at the left side.
	`WARNING`	A warning icon is displayed at the left side.
`Delay` (optional)	Timeout in milliseconds	After the specified time elapses, the Alert disappears and the previously displayed screen reappears
	`FOREVER`	Default value. The display of the Alert can be deactivated only by the user.
	Missing, negative, or invalid	Will default to `FOREVER`.

Syntax

```
<IppAlert Type="ALARM|CONFIRMATION|ERROR|INFO|WARNING"
Delay="[Number]|FOREVER">
   <Title> </Title>
   <Url> </Url>
   <Text>
      <IppPhoneNumber> </IppPhoneNumber>
   </Text>
   <Image Cache="n"> </Image>
</IppAlert>
```

Child elements

- `Title`

  The title of the Alert is displayed in the title bar.

- `Url`

  This URL is used by a `SELECT`, `UPDATE`, or `EXIT` Command, if such a Command is added to the alert.

- `Text`

  Text in an Alert. An `IppPhoneNumber` (see Section 4.16, "Phone Number") can be contained, which enables the automatic translation of a phone number stored in the local phone book into an icon, a picture clip or the name associated to the number.

- `Image`

  If an image is provided, it will replace the standard symbol for the alert. For more information about the `Image` tag, see Section 4.14, "Image". Please note that, if the `Text` element contains an `IppPhoneNumber`, the picture clip or phone type icon from the phonebook will override the image object.

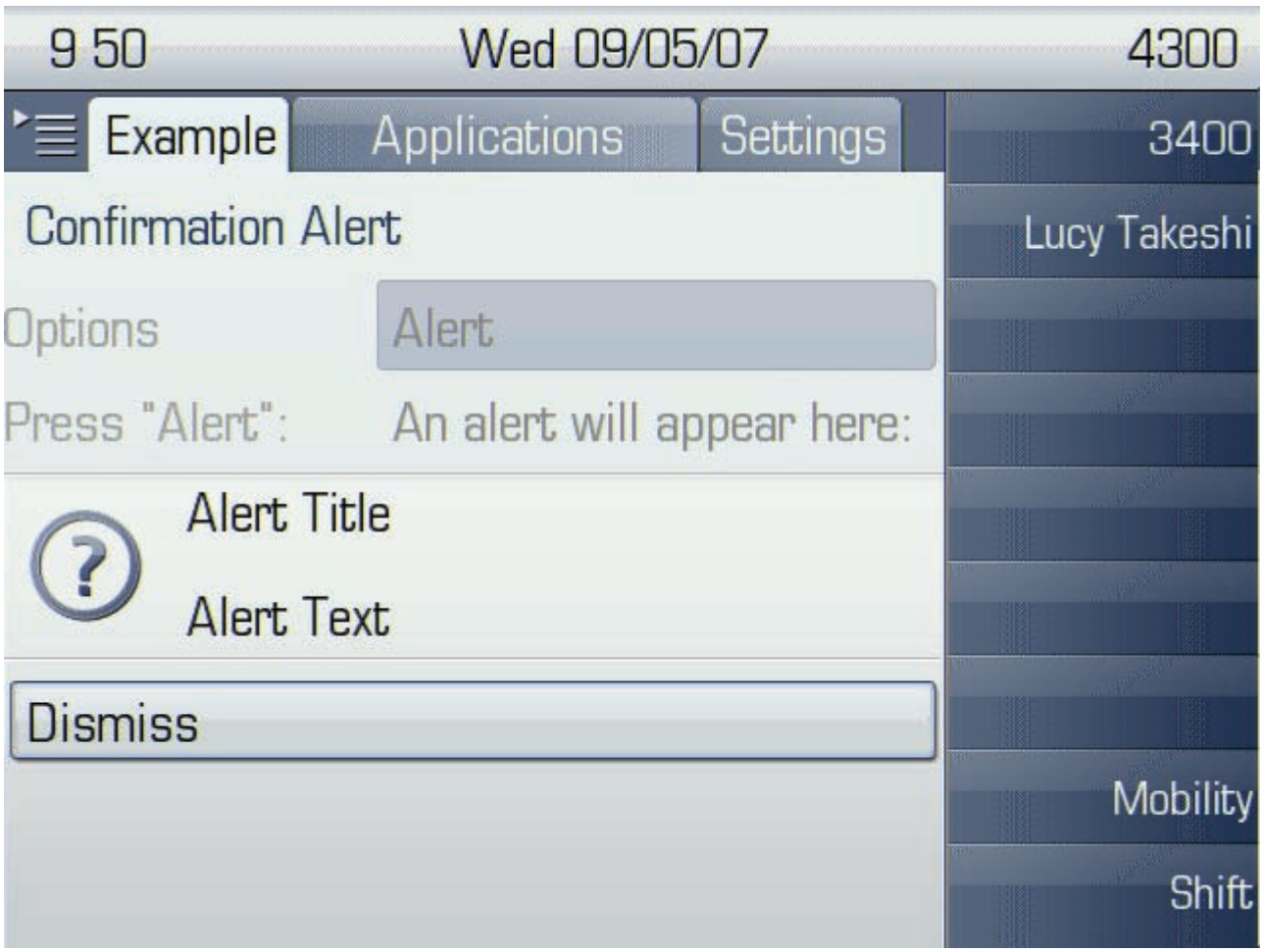
Commands

If one or more Commands are added to an Alert, i. e. to the Screen containing the Alert, these Commands will replace the implicit "Dismiss" command.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<IppPhone>
   <IppDisplay>
      <IppScreen ID="1" HiddenCount="0" CommandCount="1">
         <IppForm ItemCount="1">
            <Title>Confirmation Alert</Title>
            <Url>http://subdomain.domain/path/program</Url>
            <IppStringItem>
               <Label>Press "Alert":</Label>
               <Text>An alert will appear here:</Text>
            </IppStringItem>
         </IppForm>
         <IppCommand Type="SCREEN" Priority="0">
            <Label>Alert</Label>
            <ScreenID>2</ScreenID>
         </IppCommand>
      </IppScreen>
      <IppScreen ID="2" HiddenCount="0" CommandCount="0">
         <IppAlert Type="CONFIRMATION" Delay="FOREVER">
            <Title>Alert Title</Title>
            <Text>Alert Text</Text>
         </IppAlert>
      </IppScreen>
   </IppDisplay>
</IppPhone>
```

Screenshot


9 50
Wed 09/05/07
4300
Example
Applications
Settings
3400
Confirmation Alert
Lucy Takeshi
Options
Alert
Press "Alert":
An alert will appear here:
Alert Title
Alert Text
Dismiss
Mobility
Shift

4.9 List

This object represents a selection list, comparable to the <select> and <option> elements in HTML or WML. A Screen can only contain one List object.

There are three types of selection lists: `IMPLICIT`, `EXCLUSIVE`, and `MULTIPLE`. While the types `IMPLICIT` and `EXCLUSIVE` permit the selection of only one single option, the type `MULTIPLE` permits the selection of one or more options. Depending on the type of list, the list items are displayed as plain list (`IMPLICIT`), or with radio buttons (`EXCLUSIVE`), or checkboxes (`MULTIPLE`). Additionall, images can be added to the list items (see Screenshot).

The uniqueness of keys and values is not checked in the telephone.

If the URL indicated in the `Url` element is missing or invalid, thereby permitting no data to be sent to a server, the screen will continue to be displayed unchanged.

The following attributes are available to the element:

Attribute	Value	Function/Remarks
`Type` (mandatory)	`IMPLICIT`	Default value. As soon as an option has been selected, the data are sent immediately to the URL specified. Thus, exactly one option can be selected, and only the key/value pair that has been selected is sent to the server.
	`EXCLUSIVE`	Exactly one option can be selected, as with radio buttons. Only the key/value pair selected is sent to the server.
	`MULTIPLE`	One or more options can be selected. All key/value pairs present in the list are sent to the server. For selected options, the value of the `value` attribute is then sent; for non-selected options, this value is prefixed with `not_`.
`Count` (mandatory)	Number of options	Only as many options as indicated here are displayed. If the actual number of options exceeds this amount, all further options are ignored.
	`0`	No options are displayed.

`Columns` (optional)	Number of columns (max. 3)	A list may contain a single column or multiple columns of option text. Currently, three column lists are supported. The default value is 1, which results in a single column list. A single column list occupies 100% of the item width in a Form. The left, middle, and right columns in a three column list occupy 50%, 16%, and 34% of the item width, respectively. Each option container must have the same number of `OptionText` elements as the number of columns. The `OptionText` elements are displayed in order from the left to right column. If `Columns` is missing, negative, or invalid or there is a mismatch between the number of columns and `OptionText` elements, the list defaults to a single column list, using only the first `OptionText` element in each container.

Syntax

```
<IppList Type="IMPLICIT|EXCLUSIVE|MULTIPLE" Count="[Number]" Columns
="[Number]">
   <Title> </Title>
   <Url> </Url>
   <Option Selected="FALSE|TRUE" Key="[Key]" Value="[Value]">
      <OptionText State="GRAY|NORMAL">
         <IppPhoneNumber> </IppPhoneNumber>
       </OptionText>
      <Image Cache="[Name]"> </Image>
   </Option>
</IppList>
```

Child elements

- `Title`

  Title text for the selection list.

- `Url`

  The URL to which the data of the selected option are sent.

  If the URL is missing or is invalid, the current screen will continue to be displayed unchanged.

- `Option`

  This contains the data that are sent to the server. It can also contain descriptive text (`OptionText`) and an image (`Image`).

  The following attributes are available to the element

Attribute	Value	Function
`Selected` (optional)	`FALSE`	Default value. The option is not preselected.
	`TRUE`	The option is preselected. When the `IppList` type is `IMPLICIT`, the option is highlighted.
`Key` (optional)	Key	The key for the key/value pair that is sent to the server as an HTTP POST request. For selection lists of type `IMPLICIT` and `EXCLUSIVE`, multiple options can have the same `Key`. If the type is `MULTIPLE`, all the values must be different.
`Value` (optional)	Value	The value for the key/value pair that is sent to the server as an HTTP POST request. If the selection list type is `MULTIPLE`, all the key/value pairs are sent. For selected options, the value of the `value` attribute is then sent; for non-selected options, this value is prefixed with `not_`.

Table 4-2

If a key or value is missing or invalid, `null` is sent instead.

- `OptionText`

  Descriptive text for the option. An `IppPhoneNumber` (see Section 4.16, "Phone Number") can be included, which enables the automatic translation of a phone number stored in the local phone book into the name associated to the number, and an icon or a picture clip.

  The following attribute is available to the element

Attribute	Value	Function
`State` (optional)	`NORMAL`	The text is displayed in regular color.
	`GRAY`	The text is displayed in gray color.

- `Image`

  Illustrative image for the option. The image is displayed to the left of the text. For more information about the Image element, see Section 4.14, "Image".

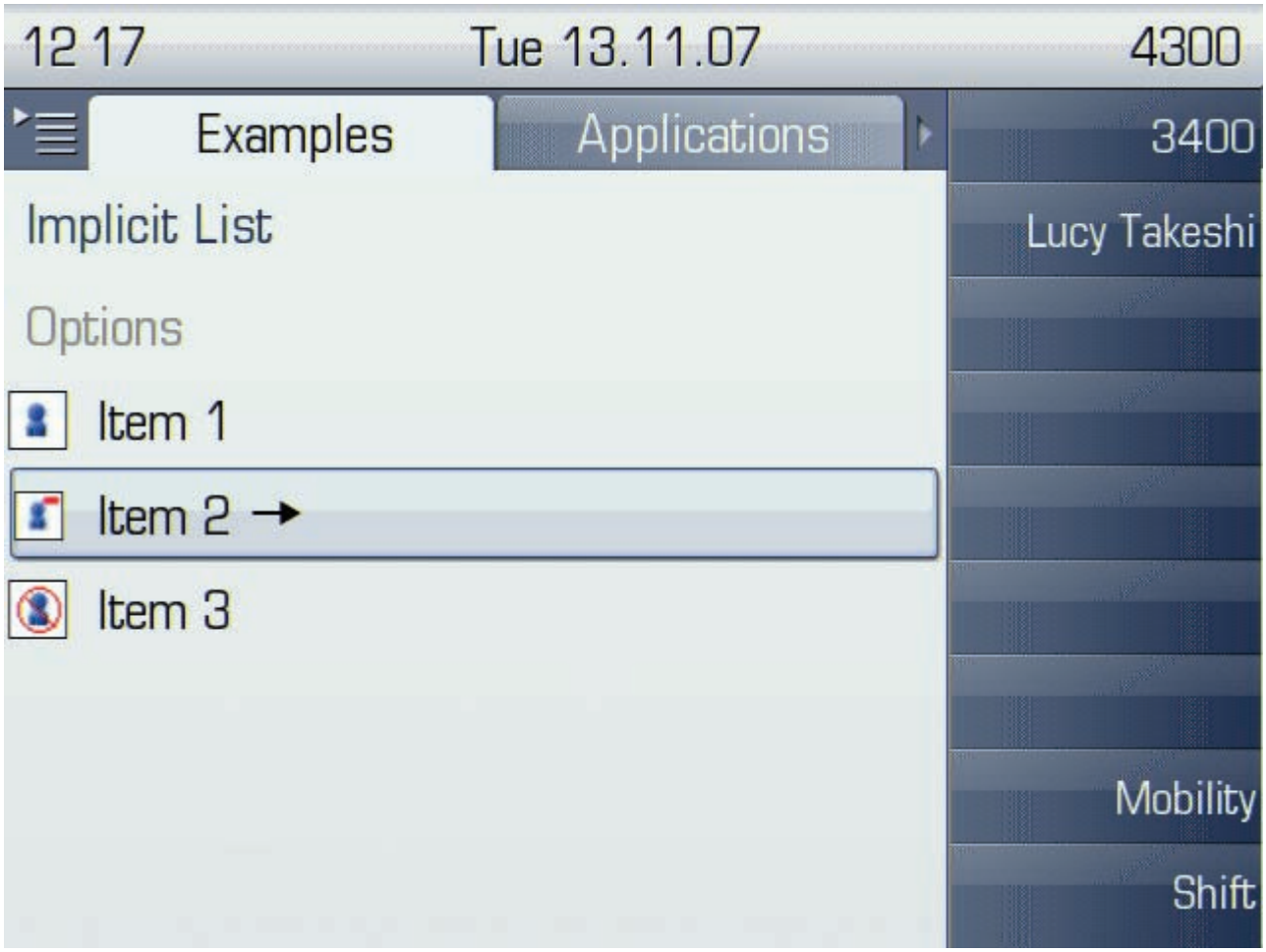
Commands

Unless the List type is `IMPLICIT`, a Command is required to send data to a server and initiate server-side action. A Command is added by inserting a Command object in the Screen (see Example). The Command can be displayed on the Options menu, which is located underneath the Screen title, or on each individual item on the List, or both. For details, please see Section 4.6, "Command".

Example

```
<?xml version="1.0" encoding="UTF-8" ?>
<IppPhone>
   <IppDisplay>
      <IppScreen ID="1" HiddenCount="0" CommandCount="1">
         <IppList Type="IMPLICIT" Count="3">
            <Title>Implicit List</Title>
            <Url>http://subdomain.domain/path/program</Url>
            <Option Selected="FALSE" Key="key1" Value="1">
               <OptionText>Item 1</OptionText>
               <Image>http://subdomain.domain/path/image1.png</Im-
age>
            </Option>
            <Option Selected="TRUE" Key="key2" Value="2">
               <OptionText>Item 2</OptionText>
               <Image>http://subdomain.domain/path/image2.png</Im-
age>
            </Option>
            <Option Selected="FALSE" Key="key3" Value="3">
               <OptionText>Item 3</OptionText>
               <Image>http://subdomain.domain/path/image3.png</Im-
age>
            </Option>
         </IppList>
         <IppCommand Type="SELECT" Priority="0">
            <Label>Select</Label>
            <ScreenID>2</ScreenID>
         </IppCommand>
      </IppScreen>
   </IppDisplay>
</IppPhone>
```

Screenshot


12 17
Tue 13.11.07
4300
Examples
Applications
3400
Implicit List
Lucy Takeshi
Options
Item 1
Item 2
Item 3
Mobility
Shift

4.10 TextBox

This element generates a text area for user input. For inputting data, the user must first navigate to the text box using the ▼ key, and then open the text editor by pressing the OK key. When the data input is completed, the user can confirm his input by pressing the ➔ key and the OK key.

Predefined content, a title for the textbox, the permitted data types (`Constraint`) as well as a server URL for data transmission can be specified.

The following attributes are available to the `IppTextBox` element:

Attribute	Value	Functionality/Remarks
`MaxSize` (optional)	Maximum number of characters	If missing, negative, or invalid, the maximum number of characters is limited only by the capacity of the display and the buffer of the phone connected.
`Constraint` (optional)	`ANY`	Default value. Any character string is permitted.
	`NUMERIC`	The data entered must be numeric. Other entries are ignored. The valid data range is +/- 2147483647.
	`PASSWORD`	The data entered are treated as a password. Asterisks (*) are displayed instead of the characters actually entered.
	`PHONENUMBER`	The data entered must be in a telephone number format.[1]
	`URL`	The data entered must represent a valid URL.[1]
	`EMAILADDR`	The data entered must represent a valid e-mail address.[1]
`Password` (optional)	`YES`	The data entered in the Text Box is confidential and should be obscured whenever possible.
	`NO`	Default.
`Default` (optional)	`NULL`	An empty Text Box is displayed; the content of the `Text` element is ignored.
	`TEXT`	The Text Box contains the content of the `Text` element.
	`PHONENUMBER`	The Text Box contains the call number of the phone.
`Key` (optional)	Key	The value of this attribute is used as a key for submitting the data as an HTTP POST request. The associated value is the content of the `Text` element. Please note that the Key must be unique within a Screen.
`Uneditable` (optional)	`YES`	The Text Box cannot be edited by the user.
	`NO`	Default value. The Text Box can be edited by the user.

1 The validation of this constraint is not implemented yet.

Syntax

```
<IppTextBox MaxSize="[Number]"
Constraint="ANY|NUMERIC|PASSWORD|PHONENUMBER|URL|EMAILADDR"
Password="YES|NO" Default="NULL|TEXT|PHONENUMBER" Key="[Key]"
Uneditable="YES|NO">
   <Title> </Title>
   <Text> </Text>
   <Url> </Url>
</IppTextBox>
```

Child elements

- `Title`

  Title of the Text Box. The title is displayed in the first line, underneath the Applications tab.

- `Text`

  Predefined text contained in the text box. This is displayed if `Default` is set to `TEXT`.

- `Url`

  URL of the server to which the data are sent.

  If the URL is missing or is invalid, thereby allowing no data to be sent to the server, the current screen will continue to be displayed unchanged.

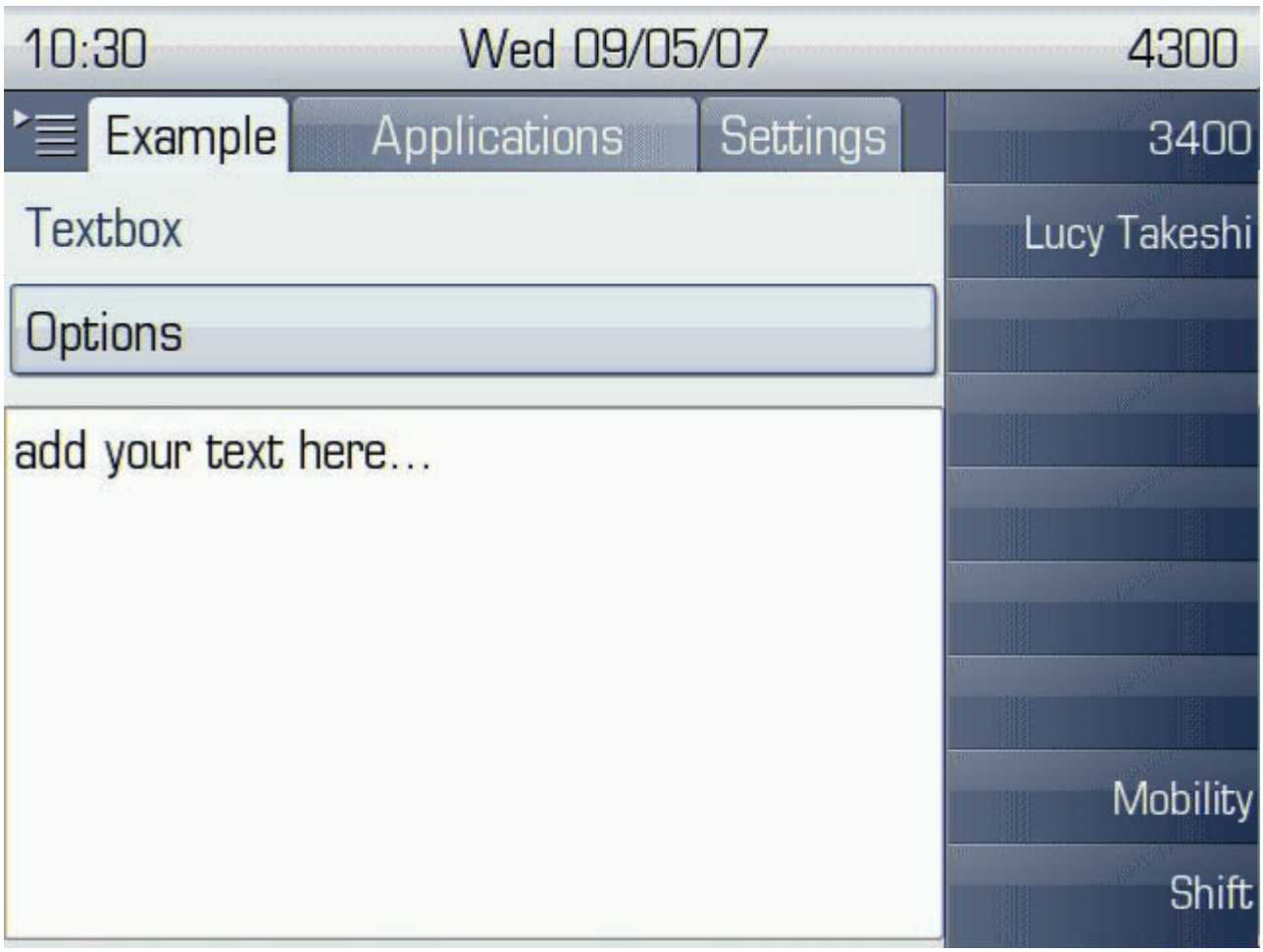
Commands

A Command is required to send the text data to a server and initiate server-side action. The Command is added by inserting a Command object in the Screen (see Example). It is displayed on the Options menu, which is located underneath the Screen title. For details about the Command object, see Section 4.6, "Command".

Example

```
<?xml version="1.0" encoding="UTF-8" ?>
<IppPhone>
   <IppDisplay>
      <IppScreen ID="1" HiddenCount="0" CommandCount="1">
         <IppTextBox MaxSize="30" Constraint="ANY" Default="TEXT"
Key="TB1">
            <Title>Textbox</Title>
            <Text>add your text here...</Text>
            <Url>http://subdomain.domain/path/program</Url>
         </IppTextBox>
         <IppCommand Type="SELECT" Priority="0">
            <Label>Select</Label>
            <ScreenID>2</ScreenID>
         </IppCommand>
      </IppScreen>
   </IppDisplay>
</IppPhone>
```

Screenshot


10:30
Wed 09/05/07
4300
Example
Applications
Settings
3400
Textbox
Lucy Takeshi
Options
add your text here...
Mobility
Shift

4.11 Ticker

Ticker text is displayed on the screen and continually moves from right to left across the display. It is positioned underneath the title bar.

Syntax

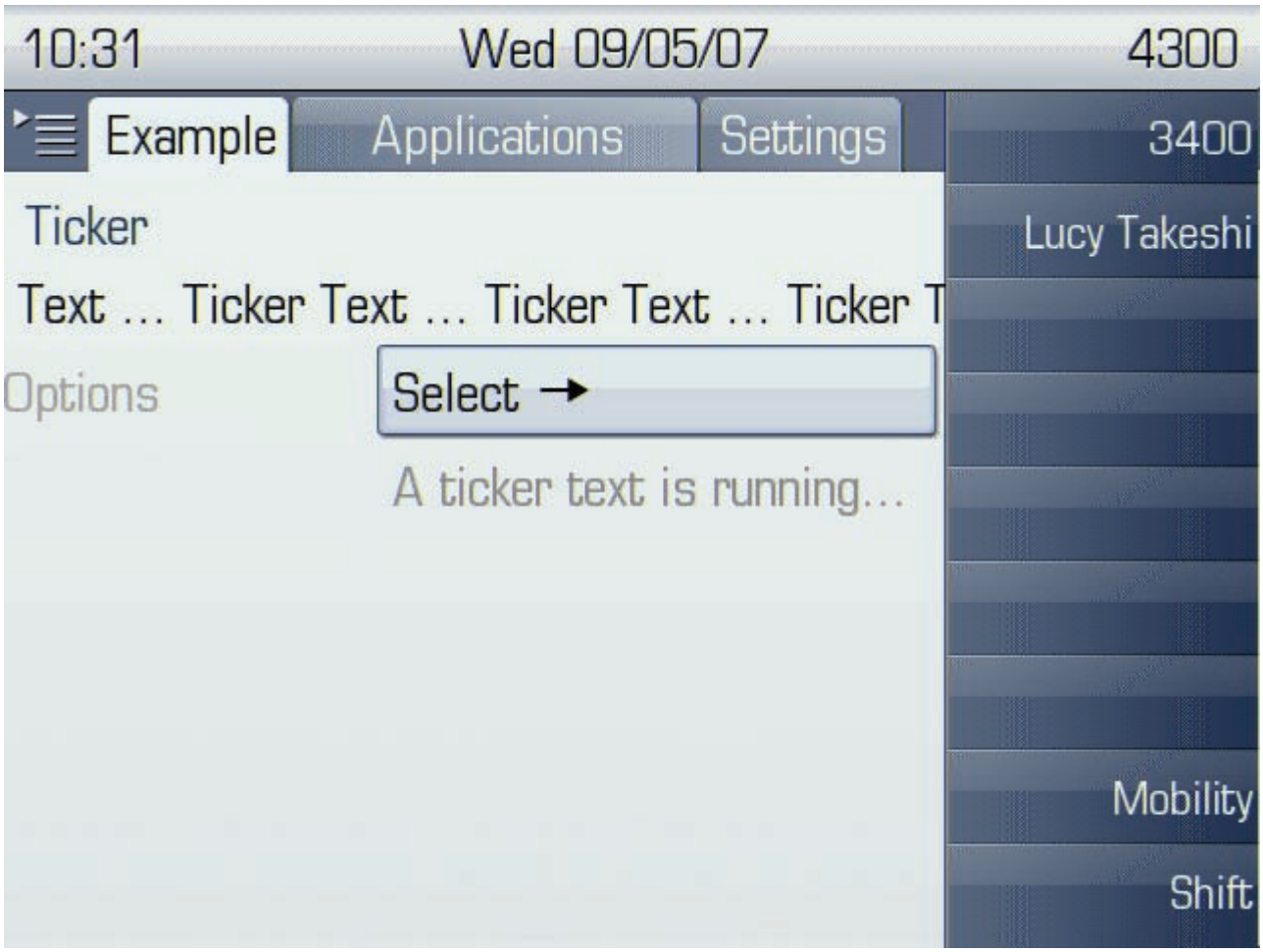
```
<IppTicker>
  <Text> </Text>
</IppTicker>
```

If the text is missing or is invalid, the ticker is not displayed at all.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<IppPhone>
   <IppDisplay>
      <IppScreen ID="1" HiddenCount="0" CommandCount="1">
         <IppForm ItemCount="1">
            <Title>Ticker</Title>
            <Url>http://subdomain.domain/path/program</Url>
            <IppStringItem>
               <Label></Label>
               <Text>A ticker text is running...</Text>
            </IppStringItem>
         </IppForm>
         <IppCommand Type="SELECT" Priority="0">
            <Label>Select</Label>
            <ScreenID>1</ScreenID>
         </IppCommand>
         <IppTicker>
            <Text>Ticker Text ... Ticker Text ...</Text>
         </IppTicker>
      </IppScreen>
   </IppDisplay>
</IppPhone>
```

Screenshot


10:31
Wed 09/05/07
4300
Example
Applications
Settings
3400
Ticker
Lucy Takeshi
Text ... Ticker Text ... Ticker Text ... Ticker T
Options
Select
A ticker text is running...
Mobility
Shift

4.12 Hidden

The function of this object is comparable to that of hidden entry fields in HTML. This makes it possible to send and receive data from the server using key/value pairs without this data being visible to the user.

The following attributes are available to the `IppHidden` element:

Attribute	Value	Function
`Type` (mandatory)	`VALUE`	Default. The value of the key/value pair is the value saved in the `Value` child element.
	`PHONENUMBER`	The value of the key/value pair is the telephone number of the phone connected.
	`IPADDRESS`	The value of the key/value pair is the IP address of the phone connected.
`Key` (mandatory)	Key	Together with the content of the `Value` child element, `Key` builds the key/value pair to be sent as an HTTP POST request. Please note that the Key must be unique within a Screen.

Syntax

```
<IppHidden Type="VALUE|PHONENUMBER|IPADDRESS" Key="[Key]">
   <Value> </Value>
</IppHidden>
```

Child elements

- `Value`

  This contains the value that is sent to the server. Together with the `Key` attribute of the `IppHidden` element, the content of `Value` builds a key/value pair.

Commands

A Command is required to send the hidden data to a server and initiate server-side action. It can be displayed on the Options menu, which is located underneath the Screen title, or, if the Screen contains a List, on each individual item on the List. If the Screen contains a Form, Commands can be displayed on individual Form elements. For details, please see Section 4.13.8, "Form/Item". For details about Commands in general, see Section 4.6, "Command".

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<IppPhone>
   <IppDisplay>
      <IppScreen ID="1" HiddenCount="1" CommandCount="1">
         <IppList Type="EXCLUSIVE" Count="3">
            <Title>Hidden</Title>
            <Url>http://subdomain.domain/path/program</Url>
            <Option ID="1" Selected="FALSE" Key="key1" Value="1">
               <OptionText>Option 1</OptionText>
            </Option>
            <Option ID="2" Selected="FALSE" Key="key2" Value="2">
               <OptionText>Option 2</OptionText>
            </Option>
            <Option ID="3" Selected="FALSE" Key="key3" Value="3">
               <OptionText>Option 3</OptionText>
            </Option>
         </IppList>
         <IppCommand Type="SELECT" DisplayOn="LISTITEM" Priority="0">
            <Label>Select</Label>
            <ScreenID>2</ScreenID>
         </IppCommand>
         <IppHidden Type="VALUE" Key="hidden key">
            <Value>hidden value</Value>
         </IppHidden>
      </IppScreen>
   </IppDisplay>
</IppPhone>
```

4.13 Form

The Form object is used to generate a frame that provides some layout options for a Screen. It provides basic information in the standard elements `Title` and `Url` and represents a container for the following elements:

- String Item (see Section 4.13.1, "Form/StringItem")
- Image Item (see Section 4.13.2, "Form/ImageItem")
- Spacer (see Section 4.13.3, "Form/Spacer")
- Text Field (see Section 4.13.4, "Form/TextField")
- Choice Group (see Section 4.13.5, "Form/ChoiceGroup")
- Date Field (see Section 4.13.6, "Form/DateField")
- Button (Section 4.13.7, "Form/Button")
- Item (Section 4.13.8, "Form/Item")
- Gauge (see Section 4.13.9, "Form/Gauge")

A Form is divided into two columns. The left column contains the "Options" label automatically created for `Command` elements, as well as any label created for an element within the form. The proportions of the columns can be modified using the `Proportions` attribute.

The following attributes and values are available to the `IppForm` element:

Attribute	Value	Function
`ItemCount` (mandatory)	Number of elements contained	Indicates the number of specific elements contained (`Title` and `Url` are not counted).
	Missing, negative or invalid	Defaults to 0. Any elements for items in the form will be ignored.
`Proportions` (optional)	`0_100`	Left column: 0% - right column: 100%
	`15_85`	Left column: 15% - right column: 85%
	`25_75`	Left column: 25% - right column: 75%
	`40_60`	Left column: 40% - right column: 60%
	`50_50`	Left column: 50% - right column: 50%
	`60_40`	Left column: 60% - right column: 40%
	`75_25`	Left column: 75% - right column: 25%

Syntax

```
<IppForm ItemCount="[Number]"
Proportions="0_100|15_85|25_75|40_60|50_50|60_40|75_25">
   <Title> </Title>
   <Url> </Url>
   <IppStringItem/>
   <IppImageItem/>
   <IppSpacer/>
   <IppTextField/>
   <IppChoiceGroup/>
   <IppDateField/>
   <IppButton/>
   <IppItem>
   <IppGauge>
</IppForm>
```

Child elements

- `Title`

  Title of the Form.

- `Url`

  URL of the server to which the data is to be sent.

If the URL indicated in the `Url` element is missing or invalid, thereby allowing no data to be sent to a server, the screen will continue to be displayed unchanged.

4.13.1 Form/StringItem

This object provides a read-only text field. The text field consists of a label and the actual text. The label is displayed in the left column, and the text is shown in the right column.

Syntax

```
<IppStringItem>
   <Label> </Label>
   <Text> </Text>
</IppStringItem>
```

Child elements

- `Label`

  Label that precedes the text.

- `Text`

  Content in the text field.

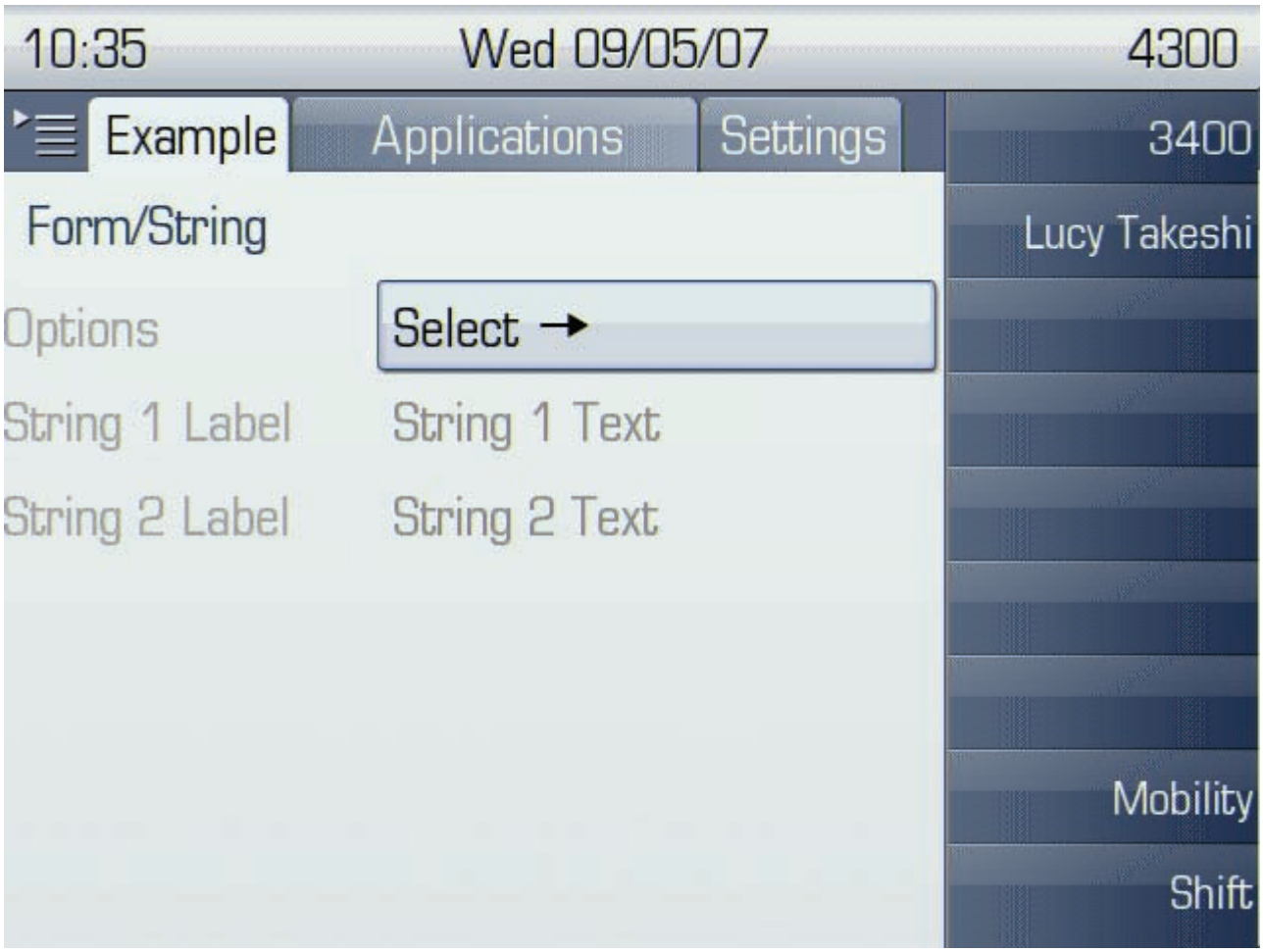
Commands

A StringItem can be made interactive by adding a Command. To achieve this, both the StringItem and the Command must be placed inside a Form Item. For details, please see Section 4.13.8, "Form/Item". For details about the Command object, see Section 4.6, "Command".

Example

```
<?xml version="1.0" encoding="UTF-8" ?>
<IppPhone>
   <IppDisplay>
      <IppScreen ID="1" HiddenCount="0" CommandCount="1">
         <IppForm ItemCount="2">
            <Title>Form/String</Title>
            <Url>http://subdomain.domain/path/program</Url>
            <IppStringItem>
               <Label>String 1 Label</Label>
               <Text>String 1 Text</Text>
            </IppStringItem>
            <IppStringItem>
               <Label>String 2 Label</Label>
               <Text>String 2 Text</Text>
            </IppStringItem>
         </IppForm>
         <IppCommand Type="SELECT" Priority="0">
            <Label>Select</Label>
            <ScreenID>2</ScreenID>
         </IppCommand>
      </IppScreen>
   </IppDisplay>
</IppPhone>
```

Screenshot


10:35
Wed 09/05/07
4300
Example
Applications
Settings
3400
Form/String
Lucy Takeshi
Options
Select →
String 1 Label
String 1 Text
String 2 Label
String 2 Text
Mobility
Shift

4.13.2 Form/ImageItem

This object functions as a frame for linking an image in the Screen layout. In case the image cannot be displayed, an alternative text is shown instead, if provided. Also, a label can be added, which is displayed in the left column.

Syntax

```
<IppImageItem>
   <Label> </Label>
   <Image Cache="[Name]"> </Image>
   <AltText> </AltText>
</IppImageItem>
```

Child elements

- `Label`

  Label for the image; it is displayed in the left column.

- `Image`

  Links the image file. For more information about the Image element, see Section 4.14, "Image".

- `AltText`

  Alternative text that is displayed if the image cannot displayed.

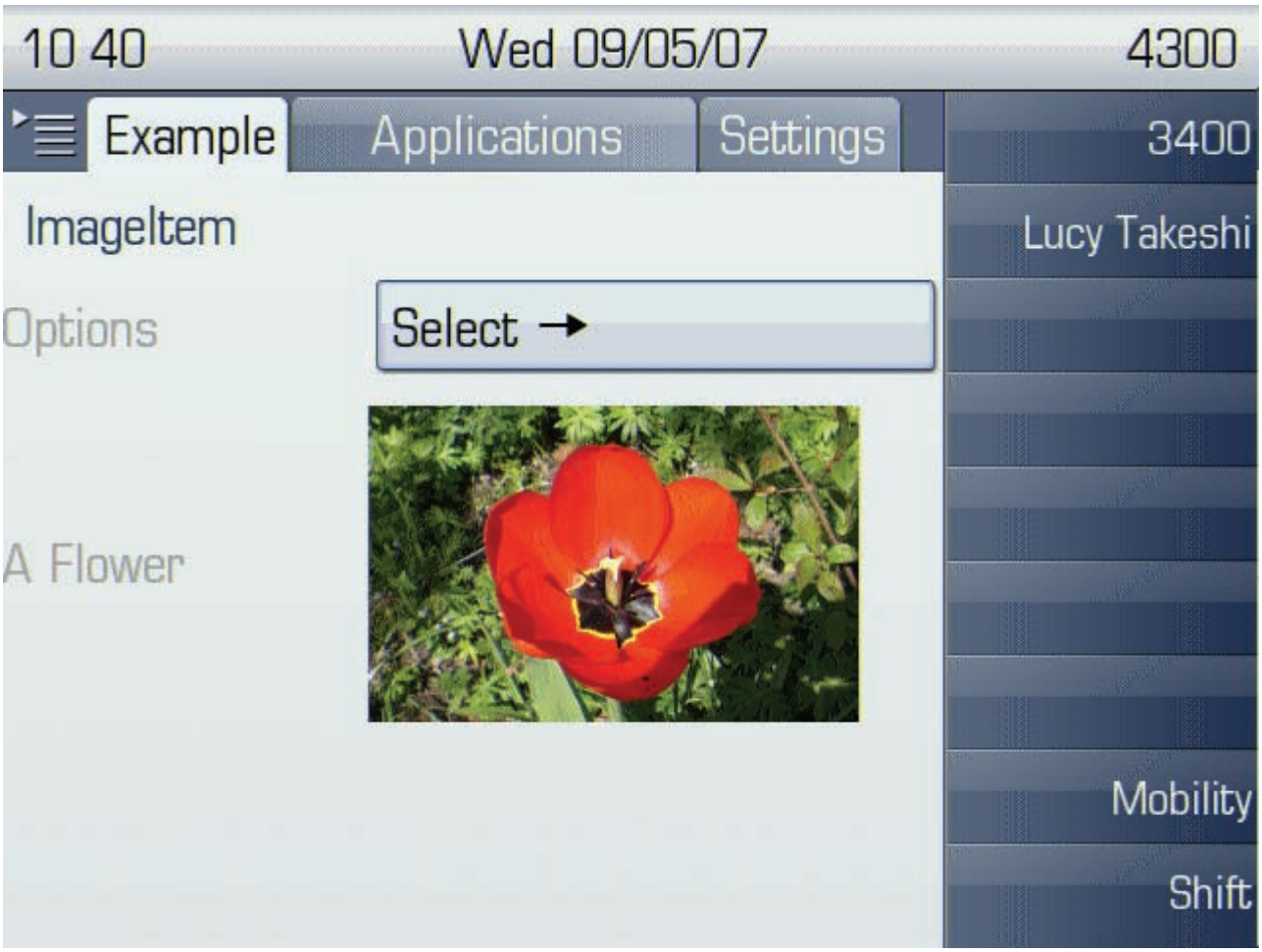
Commands

An Image Item can be made interactive by adding one or more Commands. As a specific feature of Image Items, the context menu will contain an additional Command for closing the context menu. This Command is labeled "Exit menu". To add a Command, both the Image Item and the Command must be placed inside a Form Item. For details, please see Section 4.13.8, "Form/Item". For details about the Command object, see Section 4.6, "Command"".

Example

```
<?xml version="1.0" encoding="UTF-8" ?>
<IppPhone>
   <IppDisplay>
      <IppScreen ID="1" HiddenCount="0" CommandCount="1">
         <IppForm ItemCount="1">
            <Title>ImageItem</Title>
            <Url>http://subdomain.domain/path/program</Url>
            <IppImageItem Layout="CENTER">
               <Label>A Flower</Label>
               <Image>http://subdomain.domain/path/image.png</Image>
               <AltText>The image cannot be displayed.</AltText>
            </IppImageItem>
         </IppForm>
         <IppCommand Type="SELECT" Priority="0">
            <Label>Select</Label>
             <ScreenID>2</ScreenID>
         </IppCommand>
      </IppScreen>
   </IppDisplay>
</IppPhone>
```

Screenshot


10 40
Wed 09/05/07
4300
Example
Applications
Settings
3400
ImageItem
Lucy Takeshi
Options
Select
A Flower
Mobility
Shift

4.13.3 Form/Spacer

The spacer generates an extra vertical space the height of a line between two elements.

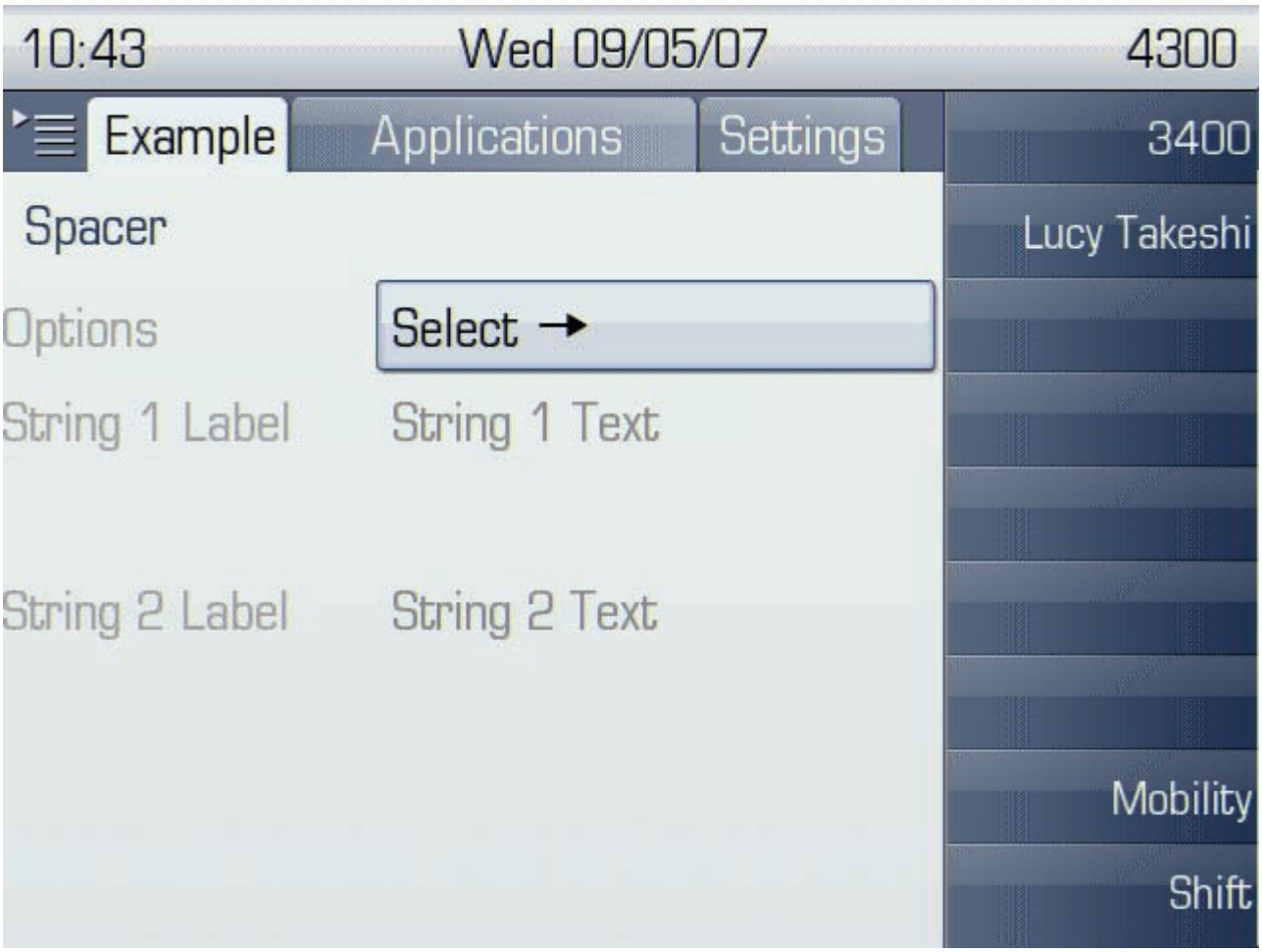
Syntax

```
<IppSpacer/>
```

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<IppPhone>
   <IppDisplay>
      <IppScreen ID="1" HiddenCount="0" CommandCount="1">
         <IppForm ItemCount="3">
            <Title>Spacer</Title>
            <Url>http://subdomain.domain/path/program</Url>
            <IppStringItem>
               <Label>String 1 Label</Label>
               <Text>String 1 Text</Text>
            </IppStringItem>
            <IppSpacer/>
            <IppStringItem>
               <Label>String 2 Label</Label>
               <Text>String 2 Text</Text>
            </IppStringItem>
         </IppForm>
         <IppCommand Type="SELECT" Priority="0">
            <Label>Select</Label>
            <ScreenID>2</ScreenID>
         </IppCommand>
      </IppScreen>
   </IppDisplay>
</IppPhone>
```

Screenshot


10:43
Wed 09/05/07
4300
Example
Applications
Settings
3400
Spacer
Lucy Takeshi
Options
Select →
String 1 Label
String 1 Text
String 2 Label
String 2 Text
Mobility
Shift

4.13.4 Form/TextField

An entry field for user provided data is generated, comparable to the <input> tag in HTML. The entry field contains a label. In addition, the data type of the entry can be defined using the `Constraint` attribute.

The following attributes are available to the `TextField` element:

Attribute	Value	Function/Remarks
`MaxSize` (optional)	Maximum number of characters	If the attribute is missing or the value is invalid, the maximum number of characters is limited by the capacity of the display and the buffer of the phone connected.
	Missing, negative, or invalid	Constrained by the display capabilities and buffers in the phone.
`Constraint` (optional)	`ANY`	Default value. Any character string is permitted.
	`NUMERIC`	The data entered must be numeric. Other keyboard entries are ignored.
	`PASSWORD`	The data entered are treated as a password. Asterisks (*) are displayed instead of the characters actually entered.
	`PHONENUMBER`	The data entered must be in a telephone number format.
	`URL`	The data entered must represent a valid URL.
	`EMAILADDR`	The data entered must represent a valid e-mail address.
`PASSWORD` (optional)	`YES`	The data entered in the text field are confidential and should be obscured whenever possible.
	`NO`	Default value. The data entered in the text field are not confidential.
`Default` (optional)	`NULL`	An empty entry field is displayed; the content of the `Text` element is ignored.
	`TEXT`	The entry field contains the content of the `Text` element.
	`PHONENUMBER`	The text field contains the phone's call number.
`Key` (optional)	Key	The value of this attribute is used as a key for submitting the data as an HTTP request. The associated value is the content of the `Text` element. Please note that the Key must be unique within a Screen.

Table 4-3

Attribute	Value	Function/Remarks
`Uneditable` (optional)	`YES`	The text field cannot be edited by the user.
	`NO`	Default value. The text field can be edited by the user.

Table 4-3

Syntax

```
<IppTextField MaxSize="[Number]" Constraint="ANY|NUMERIC|PASS-
WORD|PHONENUMBER|URL|EMAILADDR" PASSWORD="YES|NO"
Default="NULL|TEXT|PHONENUMBER" Key="[Key]" Uneditable="YES|NO">
   <Label > </Label >
   <Text> </Text>
</IppTextField >
```

Child elements

- `Label`

  Label for identifying the entry field. It is displayed in a two-column layout: the label is placed in the left column, and the entry field is placed in the right column.

- `Text`

  Text that is preset in the entry field.

Commands

A Command is required to send the data to a server and initiate server action. The Command can be displayed on the Options menu, which is located underneath the Screen title, or on the date field. To achieve this, both the field and the Command must be placed inside a Form Item.

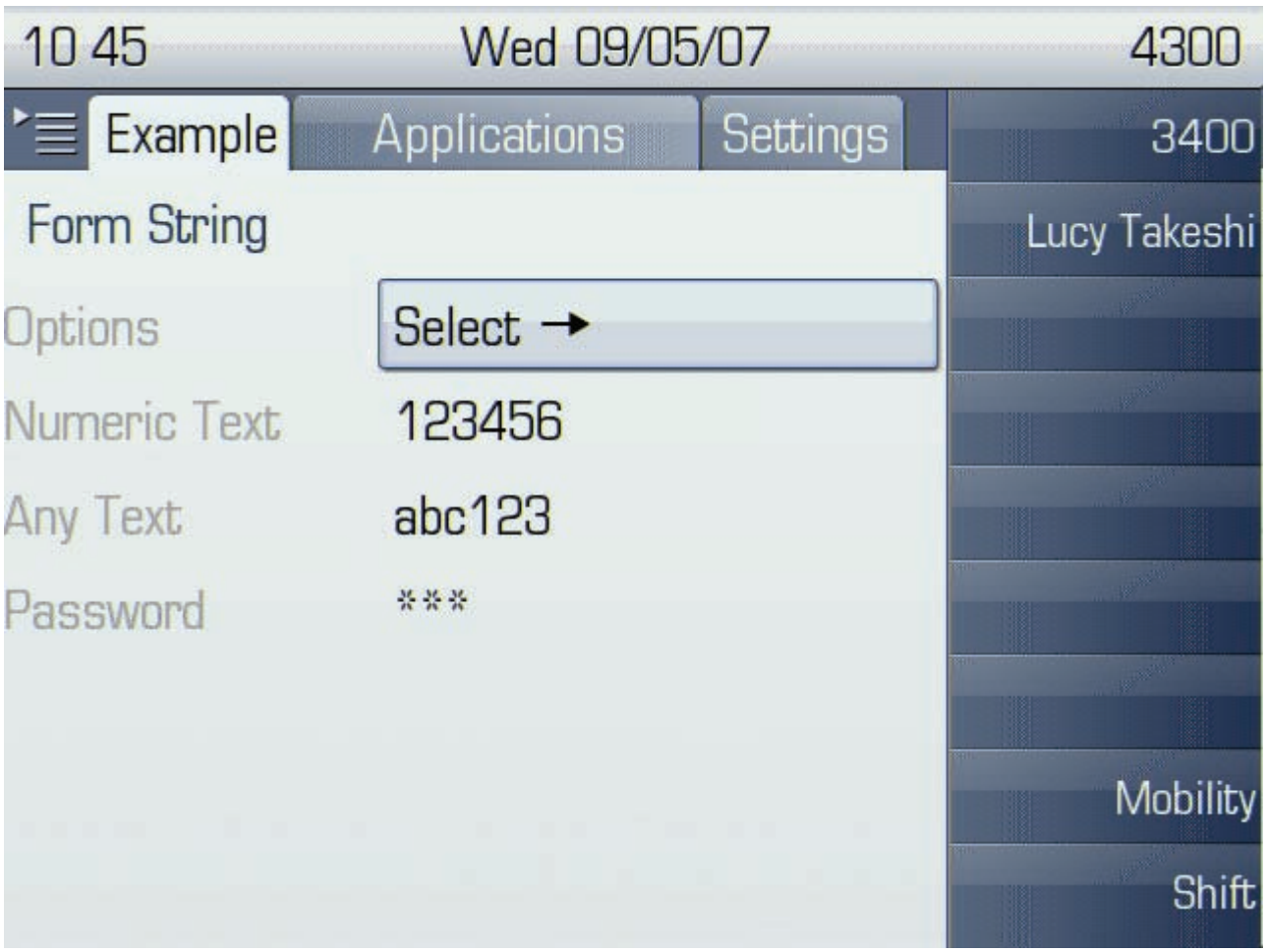
If a Command is displayed on the Text Field, an arrow is shown as soon as the field has the focus. Additionally, the TouchGuide buttons are mapped in a different way: the ➔ key now opens the context menu, so that only the OK key opens the editor.

For details, please see Section Section 4.13.8, "Form/Item". For details about the Command object, see Section 4.6, "Command".

Example

```
<?xml version="1.0" encoding="UTF-8" ?>
<IppPhone>
   <IppDisplay>
      <IppScreen ID="1" HiddenCount="0" CommandCount="1">
         <IppForm ItemCount="3">
            <Title>Form String</Title>
            <Url>http://subdomain.domain/path/program</Url>
            <IppTextField MaxSize="30" Constraint="NUMERIC" De-
fault="TEXT" Key="TF1">
               <Label>Numeric Text</Label>
               <Text>123456</Text>
            </IppTextField>
            <IppTextField MaxSize="30" Constraint="ANY" De-
fault="TEXT" Key="TF2">
               <Label>Any Text</Label>
               <Text>abc123</Text>
            </IppTextField>
            <IppTextField MaxSize="30" Constraint="PASSWORD" De-
fault="TEXT" Key="TF3">
               <Label>Password</Label>
               <Text>999</Text>
            </IppTextField>
         </IppForm>
         <IppCommand Type="SELECT" Priority="0">
            <Label>Select</Label>
            <ScreenID>2</ScreenID>
         </IppCommand>
      </IppScreen>
   </IppDisplay>
</IppPhone>
```

Screenshot


10 45
Wed 09/05/07
4300
Example
Applications
Settings
3400
Form String
Lucy Takeshi
Options
Select →
Numeric Text
123456
Any Text
abc123
Password
***
Mobility
Shift

4.13.5 Form/ChoiceGroup

This object generates a selection list within a form, comparable to the <select> and <option> elements in HTML and WML.

A Form may contain none, one, or more Choice Groups.

The uniqueness of keys and values is not checked in the telephone.

The following attributes are available to the `IppChoiceGroup` element:

Attribute	Value	Function
`Type` (mandatory)	`EXCLUSIVE`	Default value. Exactly one option can be selected (radio button function). On submit, only the key/value pair selected is sent to the server.
	`MULTIPLE`	One or more options can be selected. On submit, all the key/value pairs are sent to the server. For selected options, the value of the `value` attribute is then sent; for non-selected options, this value is prefixed with `not_`.
	`POPUP`	On startup, only the preselected option is shown. On pressing ➔, a context menu containing all options opens up. A context menu option is selected by pressing OK when it has the focus. To submit the selection made in the context menu, the `SELECT` command must be issued, just like with the other ChoiceGroup types.
`Count` (mandatory)	Number of options	Only as many options as indicated here are displayed. If the actual number of options is greater, all further options that exceed this setting are ignored.
	`0`	No options are displayed.

Syntax

```
<IppChoiceGroup Type="EXCLUSIVE|MULTIPLE|POPUP" Count="[Number]">
   <Label> </Label >
   <Option Selected="FALSE|TRUE" Key="[Key]" Value="[Value]">
      <OptionText> </OptionText>
         <IppPhoneNumber> </IppPhoneNumber>
      <Image Cache="[Name]"> </Image>
   </Option>
</ IppChoiceGroup >
```

Child elements

- `Option`

  Stores the data to be sent to the server in attributes. It can also contain descriptive text (`OptionText`) and an image (`Image`).

Attribute	Value	Function
`Selected` (optional)	`FALSE`	Default value. The option is not preselected
	`TRUE`	The option is preselected.
`Key` (optional)	Key	The key for the key/value pair that is sent to the server. For selection lists of type `POPUP` and `EXCLUSIVE`, multiple options can have the same `Key`. If the type is `MULTIPLE`, all the values must be different.
`Value` (optional)	Value	The value for the key/value pair that is sent to the server. If the selection list type is MULTIPLE, all the key/value pairs are sent. For selected options, the value of the `value` attribute is then sent; for non-selected options, this value is prefixed with `not_`.

Table 4-4

If a key or value is missing or invalid, `null` is sent instead.

- `OptionText`

  Descriptive text for the option. An `IppPhoneNumber` (see Section 4.16, "Phone Number") can be contained, which enables the automatic translation of a phone number stored in the local phone book into the name associated to the number and a picture clip or a phone type icon.

- `Image`

  Illustrative image for the option. The image is displayed in the left column. For more information about the Image object, see Section 4.14, "Image".

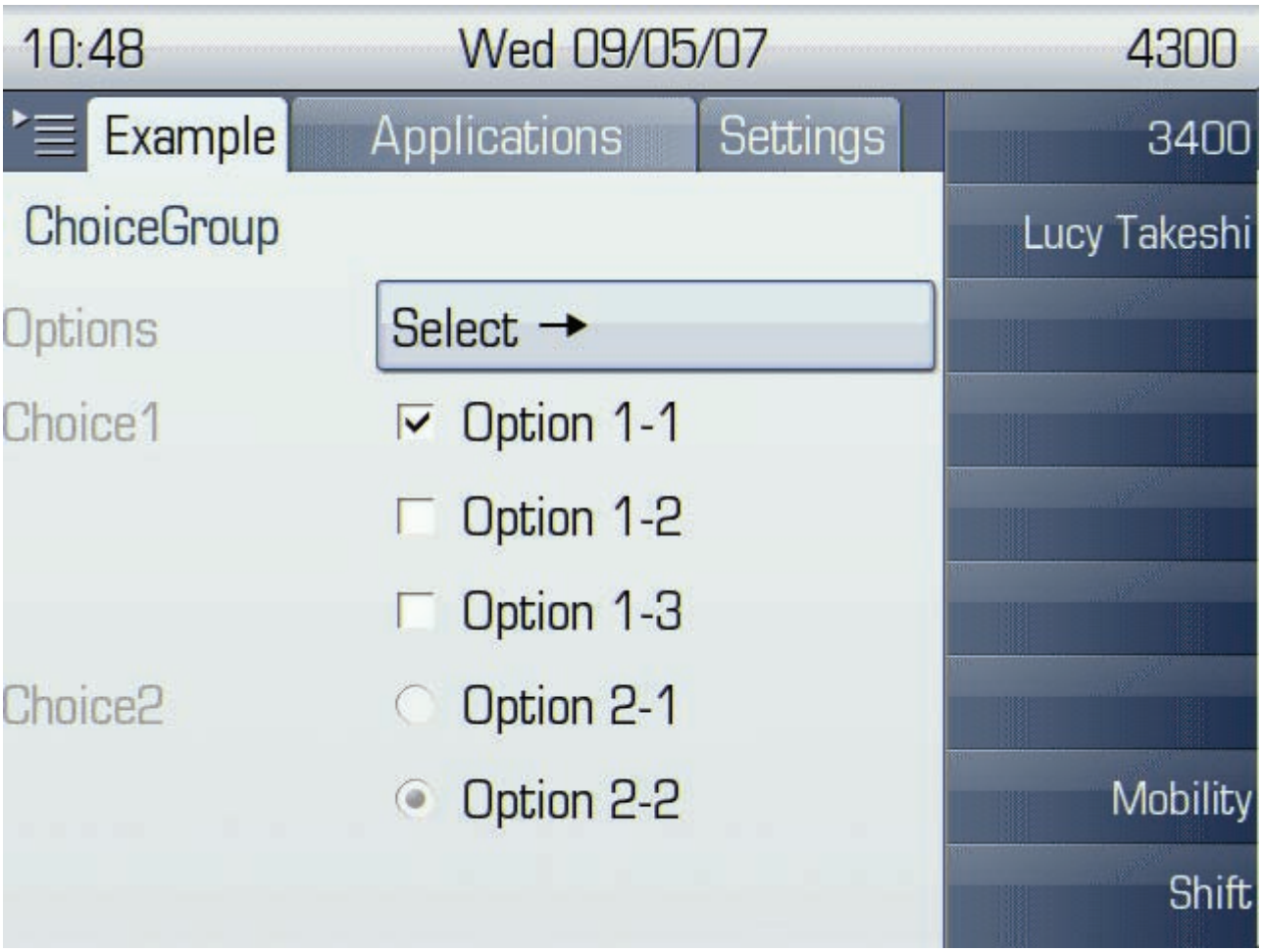
Commands

A Command is required to send the data to a server and initiate server action. The Command can be displayed on the Options menu, which is located underneath the Screen title, or on each individual Choice Group Option. To achieve this, both the Choice Group and the Command must be placed inside an Form Item. For details, please see Section 4.13.8, "Form/Item". For details about the Command object, see Section 4.6, "Command".

Example

```
<?xml version="1.0" encoding="UTF-8" ?>
<IppPhone>
   <IppDisplay>
      <IppScreen ID="1" HiddenCount="0" CommandCount="1">
         <IppForm ItemCount="2">
            <Title>ChoiceGroup</Title>
            <Url>http://subdomain.domain/path/program</Url>
            <IppChoiceGroup Type="MULTIPLE" Count="3">
               <Label>Choice1</Label>
                  <Option Selected="TRUE" Key="key1-1" Value="1">
                     <OptionText>Option 1-1</OptionText>
                     <Image></Image>
                  </Option>
                  <Option Selected="FALSE" Key="key1-2" Value="2">
                     <OptionText>Option 1-2</OptionText>
                     <Image></Image>
                  </Option>
                  <Option Selected="FALSE" Key="key1-3" Value="3">
                     <OptionText>Option 1-3</OptionText>
                     <Image></Image>
                  </Option>
            </IppChoiceGroup>
            <IppChoiceGroup Type="EXCLUSIVE" Count="2">
               <Label>Choice2</Label>
               <Option Selected="FALSE" Key="key2-1" Value="4">
                  <OptionText>Option 2-1</OptionText>
                  <Image></Image>
               </Option>
               <Option Selected="TRUE" Key="key2-2" Value="5">
                  <OptionText>Option 2-2</OptionText>
                  <Image></Image>
               </Option>
            </IppChoiceGroup>
         </IppForm>
         <IppCommand Type="SELECT" Priority="0">
            <Label>Select</Label>
            <ScreenID>2</ScreenID>
         </IppCommand>
      </IppScreen>
   </IppDisplay>
</IppPhone>
```

Screenshot


10:48
Wed 09/05/07
4300
Example
Applications
Settings
3400
ChoiceGroup
Lucy Takeshi
Options
Select
Choice1
Option 1-1
Option 1-2
Option 1-3
Choice2
Option 2-1
Option 2-2
Mobility
Shift

4.13.6 Form/DateField

This object generates a field in which date and time information can be prespecified or entered by the user. The Date Field contains a label. The data to be displayed (date, time, date and time) as well as the time zone can be specified.

On the display, the date is shown as three separate fields; day, month, and year.

The time is shown as two separate fields: hours and minutes. The user cannot enter seconds; however, seconds are returned with a default value of 00.000.

If the key in a Date Field is invalid or not present, the URL indicated in the form is opened, but the data contained in the Date Field are not sent.

The following attributes are available to the `IppDateField` element:

Attribute	Value	Function
`Mode` (optional)	`DATE`	Only the content of the `Date` field is submitted to the server.
	`TIME`	Only the content of the `Time` field is submitted to the server.
	`DATETIME`	Default value. The contents of the `Date` field and of the `Time` field are submitted to the server.
`Default` (optional)	`NULL`	Default value. The field is empty when the XML document is loaded.
	`MODE`	The contents of `Date` or `Time` are displayed when the XML document is loaded.
`DateKey` (optional)	Key for the date field	Together with the content of the `Date` field, this builds the key/value pair to be sent to the server as an HTTP request.
`TimeKey` (optional)	Key for the time field	Together with the content of the `Time` field, this builds the key/value pair to be sent to the server as an HTTP request.

Syntax

```
<IppDateField Mode="DATE|TIME|DATETIME" Default="NULL|MODE"
DateKey="[Key]" TimeKey="[Key]">
   <Label> </Label>
   <TimeZone> </TimeZone>
   <Date> </Date>
   <Time> </Time>
</IppDateField>
```

Child elements

- `Label`

  Generates a label for identifying the date and time fields. The label is displayed in the left hand column.

- `TimeZone`

  Entry for the time zone corresponding to the date and time fields.

- `Date`

  Date entry. The ordering and format for entry and display is set in the user menu, under Locality > Date format.

  For indicating the date in the XML document, the YYYY-MM-DD format is used.
  Example: `2007-11-20`.

- `Time`

  Time data is entered in the order of hours - minutes.

  For indicating the time in the XML document, the HH-MM-SS format is used, followed by a period and the milliseconds.
  Example: `13:00:00.000`.

Commands

A Command is required to send the data to a server and initiate server action. The Command can be displayed on the Options menu, which is located underneath the Screen title, or on the date field. To achieve this, both the field and the Command must be placed inside a Form Item.

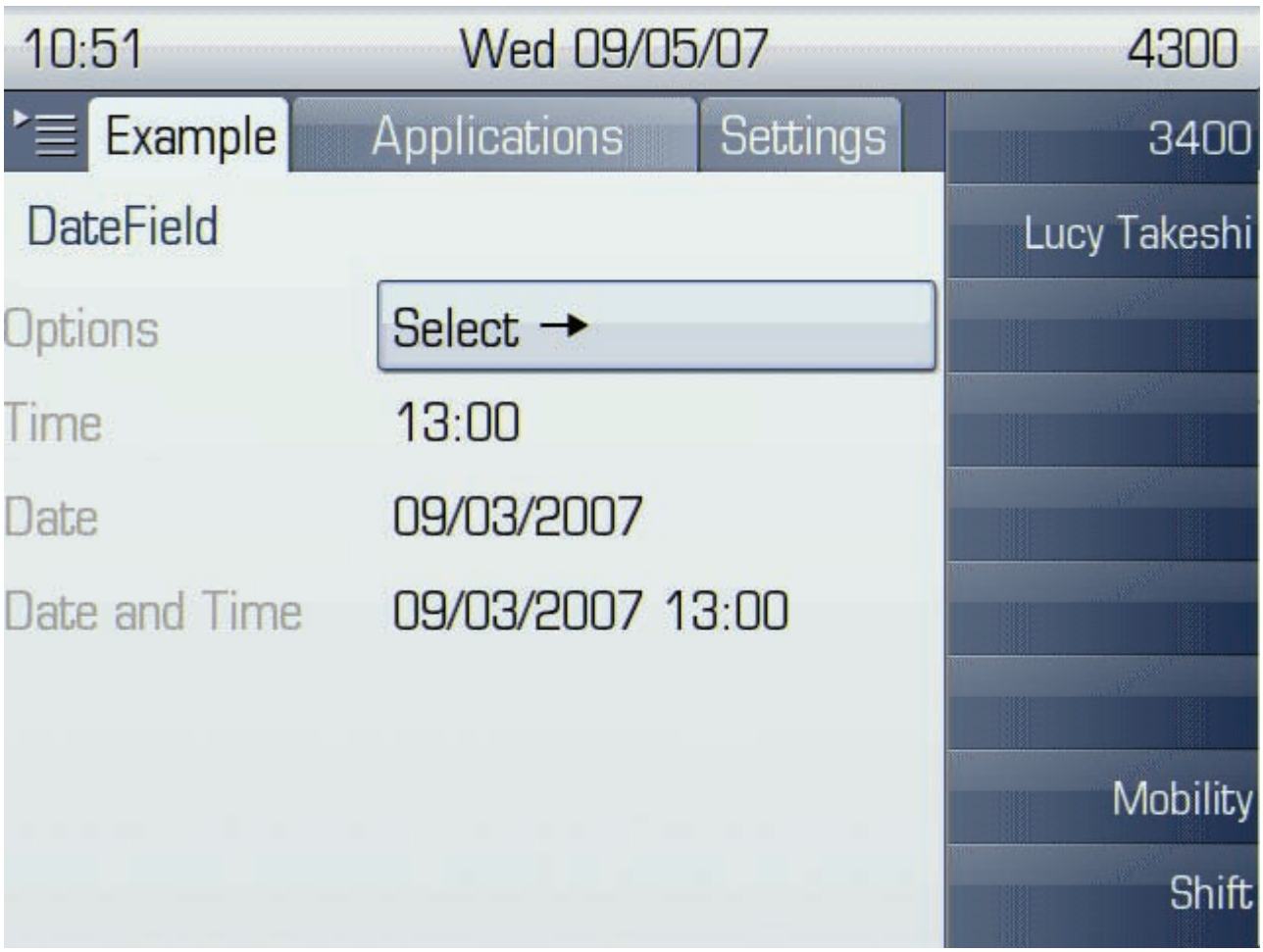
If a Command is displayed on the Date Field, an arrow is shown as soon as the field has the focus. Additionally, the TouchGuide buttons are mapped in a different way: the ➔ key now opens the context menu, so that only the OK key opens the editor.

For details, please see Section Section 4.13.8, "Form/Item". For details about the Command object, see Section 4.6, "Command".

Example

```
<?xml version="1.0" encoding="UTF-8" ?>
<IppPhone>
   <IppDisplay>
      <IppScreen ID="1" HiddenCount="0" CommandCount="1">
         <IppForm ItemCount="3">
            <Title>DateField</Title>
            <Url>http://subdomain.domain/path/program</Url>
            <IppDateField Mode="TIME" Default="MODE" DateKey="DK1"
TimeKey="TK1">
               <Label>Time</Label>
               <TimeZone>GMT+1</TimeZone>
               <Time>13:00:00.000</Time>
            </IppDateField>
            <IppDateField Mode="DATE" Default="MODE" DateKey="DK2"
TimeKey="TK2">
               <Label>Date</Label>
               <TimeZone>GMT + 1</TimeZone>
               <Date>2007-09-03</Date>
            </IppDateField>
            <IppDateField Mode="DATETIME" Default="MODE"
DateKey="DK3" TimeKey="TK3">
               <Label>Date and Time</Label>
               <TimeZone>GMT + 1</TimeZone>
               <Date>2007-09-03</Date>
               <Time>13:00:00.000</Time>
            </IppDateField>
         </IppForm>
         <IppCommand Type="SELECT" Priority="0">
            <Label>Select</Label>
            <ScreenID>2</ScreenID>
         </IppCommand>
      </IppScreen>
   </IppDisplay>
</IppPhone>
```

Screenshot


10:51
Wed 09/05/07
4300
Example
Applications
Settings
3400
DateField
Lucy Takeshi
Options
Select →
Time
13:00
Date
09/03/2007
Date and Time
09/03/2007 13:00
Mobility
Shift

4.13.7 Form/Button

The Button object generates a clickable area. When clicked, the key/value pair contained in the attributes of the `IppButton` element as well as the key/value pairs of any Hidden objects in the XML document (see Section 4.12, "Hidden") are immediately sent to the server. The URL of the server-side program is indicated in the `Form` element.

Adjacent buttons are placed vertically.

The following attributes are available to the `IppButton` element:

Attribute	Value	Function
`Type` (optional)	`IMAGE`	Default value. In the current versions, this is the only possible value. The button is represented by an image.
`Key` (mandatory)	Key	The key for the key/value pair that is sent to the server as an HTTP request. Please note that the Key must be unique within a Screen.
`Value` (mandatory)	Value	The value for the key/value pair that is sent to the server as an HTTP request.

Table 4-5

Syntax

```
<IppButton Type="IMAGE" Key="[Key]" Value="[Value]">
   <Label> </Label>
   <Image Cache="n"> </Image>
</IppButton>
```

Child elements

- `Label`

  Generates a label for identifying the button.

- `Image`

  Links an image file. The image is used as a clickable area.

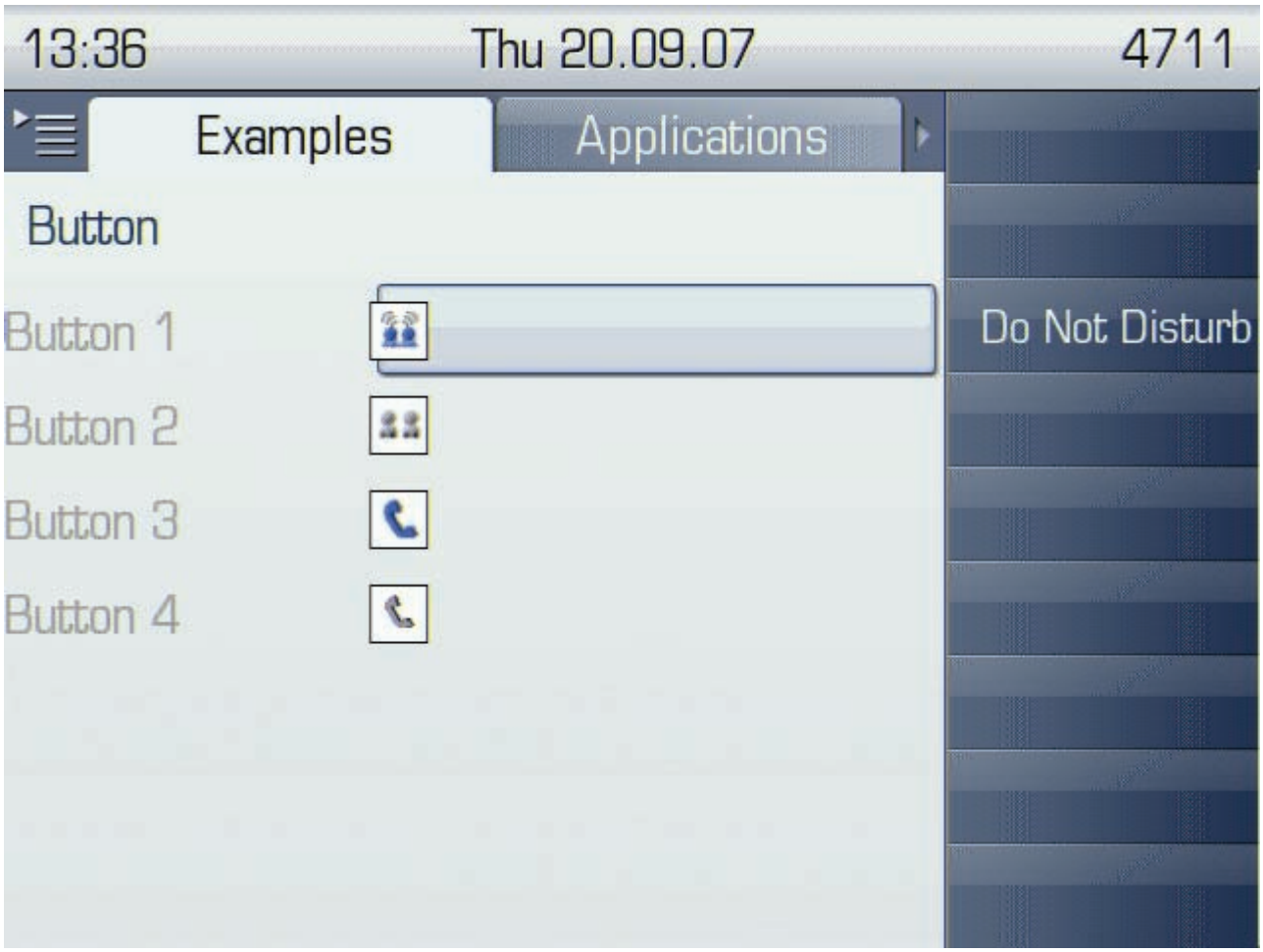
Commands

If an additional Command is added to a Button using a Form Item (see Section 4.13.8, "Form/Item"), an extra option labeled Select is inserted into the context menu.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<IppPhone>
   <IppDisplay>
      <IppScreen ID="1" HiddenCount="0" CommandCount="0">
         <IppForm ItemCount="4">
            <Title>Button</Title>
            <Url>http://subdomain.domain/path/program</Url>
            <IppButton Type="IMAGE" Layout="CENTER" Key="key1"
Value="1">
               <Label>Button 1</Label>
               <Image>http://subdomain.domain/path/image1.png</Im-
age>
            </IppButton>
            <IppButton Type="IMAGE" Layout="CENTER" Key="key2"
Value="2">
               <Label>Button 2</Label>
               <Image>http://subdomain.domain/path/image2.png</Im-
age>
            </IppButton>
            <IppButton Type="IMAGE" Layout="CENTER" Key="key3"
Value="3">
               <Label>Button 3</Label>
               <Image>http://subdomain.domain/path/image3.png</Im-
age>
            </IppButton>
            <IppButton Type="IMAGE" Layout="CENTER" Key="key4"
Value="4">
               <Label>Button 4</Label>
               <Image>http://subdomain.domain/path/image4.png</Im-
age>
            </IppButton>
         </IppForm>
      </IppScreen>
   </IppDisplay>
</IppPhone>
```

Screenshot


13:36
Thu 20.09.07
4711
Examples
Applications
Button
Button 1
Button 2
Button 3
Button 4
Do Not Disturb

4.13.8 Form/Item

This element enables to add Commands to individual items in a Form by grouping them as child elements. When the user navigates to a Form Item and presses the ➔ key on the TouchGuide, a context menu containing the Commands is opened.

The following objects can be contained in an Item:

- String Item (see Section 4.13.1, "Form/StringItem")
- Image Item (see Section 4.13.2, "Form/ImageItem")
- Spacer (see Section 4.13.3, "Form/Spacer")
- Text Field (see Section 4.13.4, "Form/TextField")
- Choice Group (see Section 4.13.5, "Form/ChoiceGroup")
- Date Field (see Section 4.13.6, "Form/DateField")
- Button (see Section 4.13.7, "Form/Button")
- Gauge (see Section 4.13.9, "Form/Gauge")

The following attribute is available to the `IppItem` element:

Attribute	Value	Function/Remarks
`CommandCount` (mandatory)	Number of Commands within the `IppItem`	The Commands are rendered correctly.
	`0`	Any `IppCommand` elements will be ignored.
	Missing, negative or invalid	The value will default to 0.
	Less than the number of Commands within the `IppItem`	Only those Commands that fall within the given number are displayed.

Syntax

```
<IppItem CommandCount="[Number of Commands]">
   <IppStringItem/>
   <IppImageItem/>
   <IppSpacer/>
   <IppTextField/>
   <IppDateField/>
   <IppChoiceGroup/>
```

```
   <IppGauge/>
   <IppButton/>
</IppItem>
```

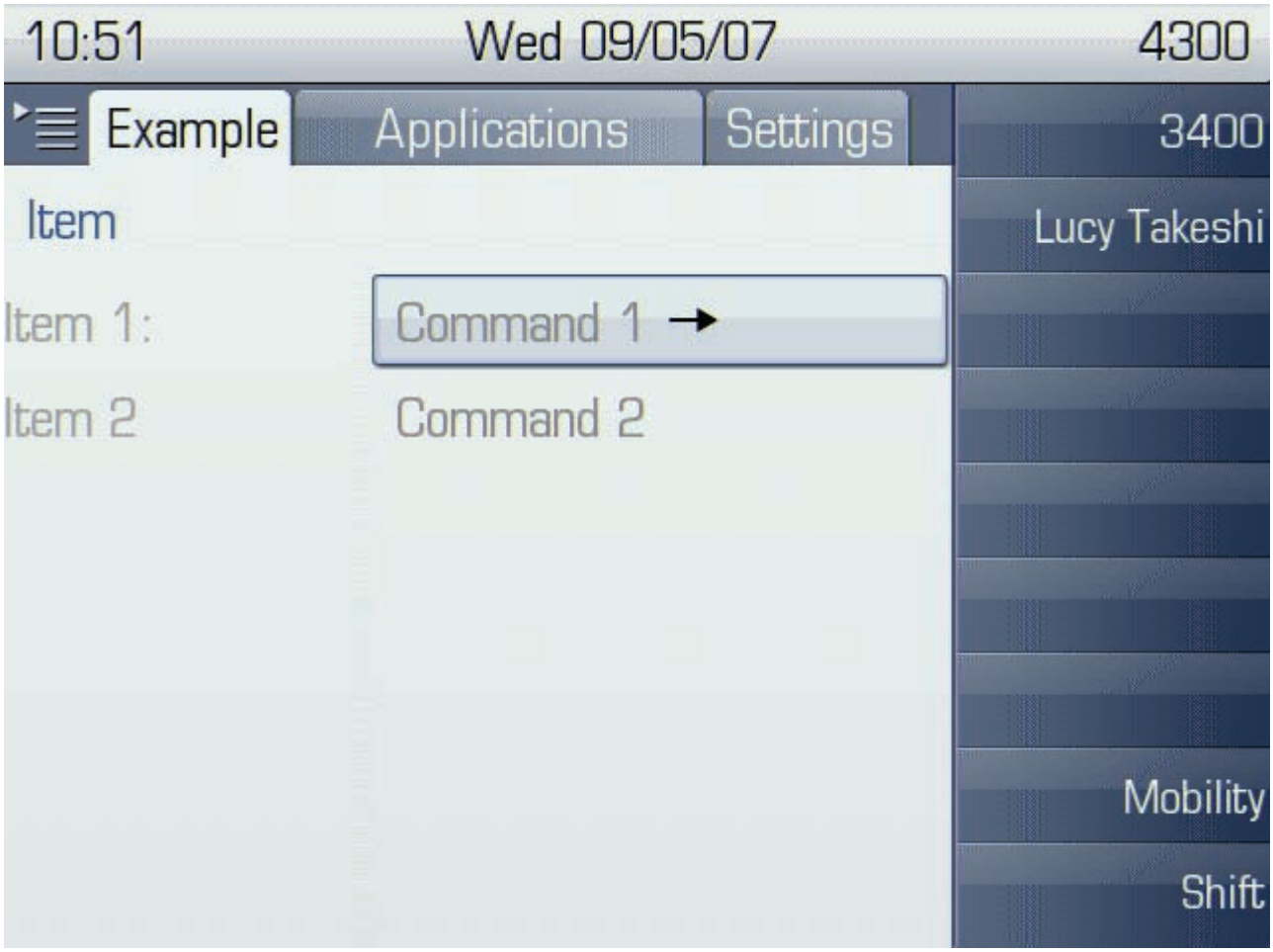
Commands

To add a Command to a Form Item, just include an `IppCommand`. The `DisplayOn` attribute must be set to `LISTITEM` or to `BOTH`. In the latter case, all Commands assigned to Form Items are available from the Options menu as well. Note that the Command's Label functions as an identifier; if two or more Commands added to Form Items have the same label, only one Command will be displayed on the Options menu.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<IppPhone>
   <IppDisplay>
      <IppScreen ID="1" HiddenCount="1" CommandCount="2">
         <IppForm ItemCount="2">
            <Title>Item</Title>
            <Url>http://subdomain.domain/path/program</Url>
            <IppItem CommandCount="1">
               <IppStringItem>
                  <Label>Item 1:</Label>
                  <Text>Command 1</Text>
               </IppStringItem>
               <IppCommand Type="SELECT" DisplayOn="LISTITEM" Priori-
ty="0">
                  <Label>Select</Label>
                  <ScreenID>1</ScreenID>
               </IppCommand>
            </IppItem>
            <IppItem CommandCount="1">
               <IppStringItem>
                  <Label>Item 2</Label>
                  <Text>Command 2</Text>
               </IppStringItem>
                <IppCommand Type="SELECT" DisplayOn="LISTITEM" Prior-
ity="1">
                  <Label>Select</Label>
                  <ScreenID>1</ScreenID>
               </IppCommand>
            </IppItem>
         </IppForm>
      </IppScreen>
   </IppDisplay>
</IppPhone>
```

Screenshot


10:51
Wed 09/05/07
4300
Example
Applications
Settings
3400
Item
Lucy Takeshi
Item 1:
Command 1
Item 2
Command 2
Mobility
Shift

4.13.9 Form/Gauge

A value or time interval is represented graphically using a horizontal status bar. Information on what is shown on the gauge can be given in the label.

Depending on what is specified, the Gauge can function as slider or as status bar. For the former, the value is set by the user and sent to a server. For the latter, the value is dependent on processes which last a longer period of time, and the bar automatically grows from the left to the right.

The following attributes are available to the `IppGauge` element:

Attribute	Value	Function
`Interactive` (optional)	`USER`	Default value. The user can change the value; the changes are graphically represented by the bar. The interactive gauge has a pre-defined value range between 0 and 100.
	`AUTO`	The user has no influence on the graphical bar. The gauge increments every second. The range, and, consequently, the time interval, is specified by the content of the `Maximum` element and by the content of the `Initial` element.
`Key` (optional)	Key for the current value	Only effective if the value of the `Interactive` attribute is `USER`. Together with the value currently set (value range 1-100), it forms a key/value pair that can be sent to a server as an HTTP request. If the key is invalid or not present, the URL specified is opened, but no data are sent to the server. For sending data to the server, the Screen must contain a `SELECT` command. Please note that the key must be unique within a Screen.

Syntax

```
<IppGauge Interactive="USER|AUTO" Key="k">
   <Label> </Label>
   <Maximum> </Maximum>
   <Initial> </Initial>
</IppGauge>
```

Child elements

- `Label`

  The label for giving information on what is displayed by the gauge.

- `Maximum`

  Data format and function are differentiated depending on the value of the `Interactive` attribute:

  - If `INTERACTIVE="USER"`: The number entered here indicates the maximum value that can be sent to the server. The range is 0-100.

  If the maximum value is set at under 100, for instance, at 80, the slider bar will still reach to the right end of the scale. If the user sets the slider to this position, the specified maximum value will be sent to the server.

  If the maximum value is missing, negative, invalid, or greater than 100, it will default to 100. If the initial value is greater than the maximum value, it will be set to the maximum value.

  - If `INTERACTIVE="AUTO"`: If the type is non-interactive, the time duration is indicated until the status bar has reached the right end. The value is given in mm:ss or in ss format, like, for instance, `01:20` for one minute and 20 seconds. The range is not limited.

  If the maximum value is missing, negative, or invalid, it will default to 100 seconds. If the initial value is greater than the maximum value, it will be set to the maximum value and there will be no automatic updates to the gauge.

- `Initial`

  Data format and function are differentiated depending on the value of the `Interactive` attribute:

  - If `INTERACTIVE="USER"`: The number indicated here determines the initial value and, therefore, the starting position of the slider bar. The range is 0-100. If the value is 20, for instance, the status bar is displayed at 20% of its total width. If the value is 0, the bar is on the leftmost position.
  - If `INTERACTIVE="AUTO"`: The number indicated here determines the starting position of the status bar. The value is given in mm:ss or in ss format.

If the initial value is missing, negative, or invalid, it will default to 0.

Commands

A Command is required to send the Gauge data to a server and initiate server-side action. The Command can be displayed on the Options menu, which is located underneath the Screen title, or on the Gauge. To achieve this, both the Gauge object and the Command must be placed inside a Form Item.

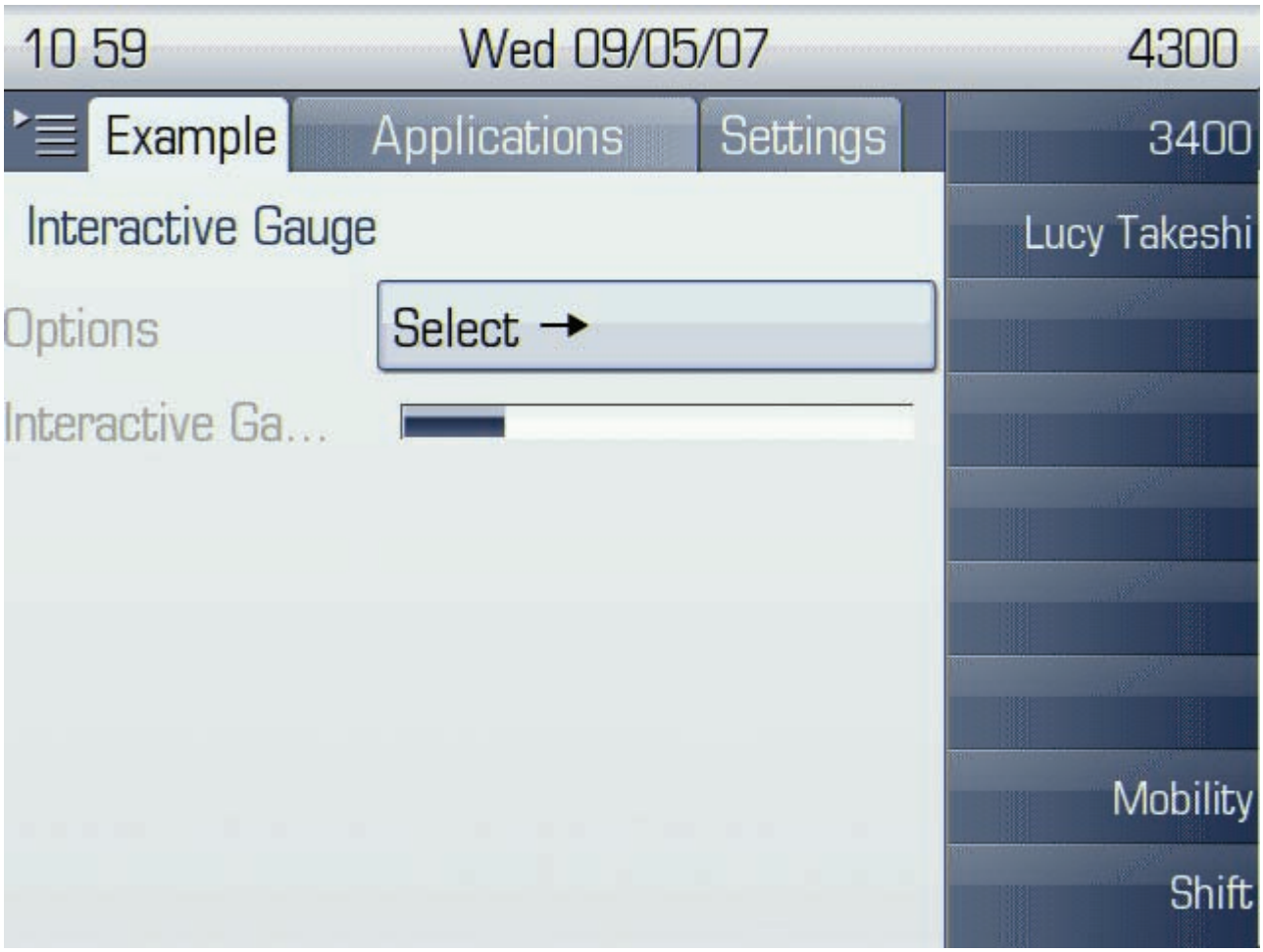
If a Command is displayed on the Gauge, the TouchGuide buttons are mapped in a different way: the ➔ key now opens the context menu. Thus the user must press the OK key first in order to change the value of an interactive gauge.

For details, please see Section Section 4.13.8, "Form/Item". For details about the Command object, see Section 4.6, "Command".

Example

```
<?xml version="1.0" encoding="UTF-8" ?>
<IppPhone>
   <IppDisplay>
      <IppScreen ID="1" HiddenCount="0" CommandCount="1">
         <IppForm ItemCount="1">
            <Title>Interactive Gauge</Title>
            <Url>http://subdomain.domain/path/program</Url>
            <IppGauge Interactive="USER" Key="G1">
               <Label>Interactive Gauge</Label>
               <Maximum>100</Maximum>
               <Initial>20</Initial>
            </IppGauge>
         </IppForm>
         <IppCommand Type="SELECT" Priority="0">
            <Label>Select</Label>
            <ScreenID>2</ScreenID>
         </IppCommand>
      </IppScreen>
   </IppDisplay>
</IppPhone>
```

Screenshot


10 59
Wed 09/05/07
4300
Example
Applications
Settings
3400
Interactive Gauge
Lucy Takeshi
Options
Select
Interactive Ga...
Mobility
Shift

4.14 Image

The Image element is used to link an image. The XML document specifies whether the image should be cached or not. The image is downloaded from the server via HTTP, analogous to images in HTML pages.

Saving images in cache is explicitly indicated in the XML document. In addition to the source of the image file, the cache name must be defined as well. This is necessary because specific areas in the cache are overwritten when sufficient memory is not available and new images are loaded. When this occurs, the previously cached images that have been overwritten must be reloaded.

The following attributes are available to the `Image` element:

Attribute	Value	Function
`Cache` (optional)	Name	The image can be accessed in cache under the name indicated here. If the attribute is invalid or not present, it is not cached.

Syntax

```
<Image Cache="[Name]">[source]</Image>
```

The content of the Image element indicates the source of the image to be displayed. Depending on the situation, this path may have various formats:

The source path of an image may take one of the following formats:

- URL for the image file: `http://<IP address>:<Port>/<filename>`
- URL of a program on the server that sends the desired image file; the key/value pair identifies the image:
  `http://<IP address>:<Port>/<filename of the program>?<key/value pair>`

If the source of the image file is missing, but the Cache attribute exists and the image is cached, the image in cache is displayed.

Images as Icons

If an image is to be used as an icon for an interactive element, it is recommended to choose an appropriate size. The images should be in PNG format. The following list provides size information about the built-in icons for OpenStage 60/80:

- Icons for mode keys, call states, phone number identification, presence, media widget, speller, idle screen, info bar, and others: 36px x 36px.

- Icons in popups and prompts: 60px x 60px.
- Dial phone number icon: 12px x 36px.

Generally, images should be optimized for OpenStage 80 phones. On OpenStage 60, they will be resized correctly.

4.15 Player

The Player is a special extension designed for applications that involve audio streaming. The XML document requests the setting up of a call for voice streaming and the displaying of the player from the server. All player commands issued by the user are sent to the server, which controls the streaming.

The Player is a transient screen, replacing the current screen on the display. However, those parts of the current screen which are not covered by the player are still visible in the background. The player displays the current progress through a media stream, and offers the functions Play, Pause, Stop, Rewind, Skip rewind, Forward, Skip forward, Call and Delete, similar to existing audio players.

The user interface consists of a gauge and a number of buttons. The gauge may be interactive or non-interactive. The gauge value is automatically updated on the phone, providing a time-based maximum value is supplied. For further information, please refer to Section 4.13.9, "Form/Gauge".

The player has eight modes of operation, which dictates the state, behaviour, and appearance of the player: Playing, Edit, Stopped playing, Recording, Stopped record, Playback recording, Edit playback, and Disabled. The Disabled mode is controlled locally on the phone, whereas the other seven modes are controlled by the remote program.

Additionally, three states are defined for the player: `PLAYING`, `RECORDING`, and `STOPPED`. The modes dictate which function buttons are displayed and their state, the default highlighting, and the type of gauge and its interaction.

The player buttons and the TouchGuide allow user interaction with the server-side program. Whenever the user presses a TouchGuide or player button, a specific key/value pair, consisting of the player's `Key` attribute and the function type, is immediately sent to the specified URL. If, for instance, the value of the `Key` attribute is `MyPlayer`, and the user presses the forward button, the key/value pair is `MyPlayer=forward`. Also the key/value pairs of the gauge (see Section 4.13.9, "Form/Gauge") and of any hiddens (see Section 4.12, "Hidden") are sent to the URL.

The following table shows the buttons and the related function types, i. e. the value of the key/value pair whose key is the value of `Key`:

Player Button	Function
	`delete`
	`rewind`
	`skip_rewind`
	`play`
	`pause`
	`stop`
	`skip_forward`
	`forward`
	`call`
	`confirm`

Table 4-6

Player Button	Function
	`record`

Table 4-6

The functions controlled by the TouchGuide are listed in the following table:

TouchGuide Button	Function
	next
(Long Press)	first
	previous
(Long Press)	last
	cancel
(Long Press)	reset

The following attributes are available to the `IppPlayer` element:

Attribute	Value	Function/Remarks
`Mode` (mandatory)	`CALL`	With this value, the phone can be in Playing, Edit, and Stopped mode.
	`RECORD`	With this value, the phone can be in Recording, Stopped Record, Edit Playback, and Playback Recording mode.
`State` (mandatory)	`PLAYING`	All buttons are displayed except for the Play button, and the gauge is automatically updating.
	`STOPPED`	All buttons are displayed except for the Stop button, and the gauge is not automatically updating.
	`RECORDING`	Only the Stop and the Delete button are displayed, and the gauge is automatically updating.
`Key` (mandatory)	Key for the Player	Together with the function type of the player button that has been pressed, the key forms a key/value pair. Please note that the Key must be unique within a Screen.

The following table shows the possible Player modes and their corresponding Interactive, State, and Mode values:

Player Mode	Attributes		
	Gauge - Interactive	**State**	**Mode**
Playing	`AUTO`	`PLAYING`	`CALL`
Edit	`USER`	`STOPPED`	`CALL`
Stopped	`AUTO`	`STOPPED`	`CALL`
Recording	`AUTO`	`RECORDING`	`RECORD`
Stopped Record	`AUTO`	`STOPPED`	`RECORD`
Edit Playback	`USER`	`STOPPED`	`RECORD`
Playback Recording	`AUTO`	`PLAYING`	`RECORD`

Table 4-7

Syntax

```
<IppPlayer State="PLAYING|STOPPED|RECORDING" Mode="CALL|RECORD"
Key="[Key]">
   <Url> </Url>
   <IppGauge> </IppGauge>
</IppPlayer>
```

Child Elements

- `Url`

  The URL invoked when a button is pressed.

- `IppGauge`

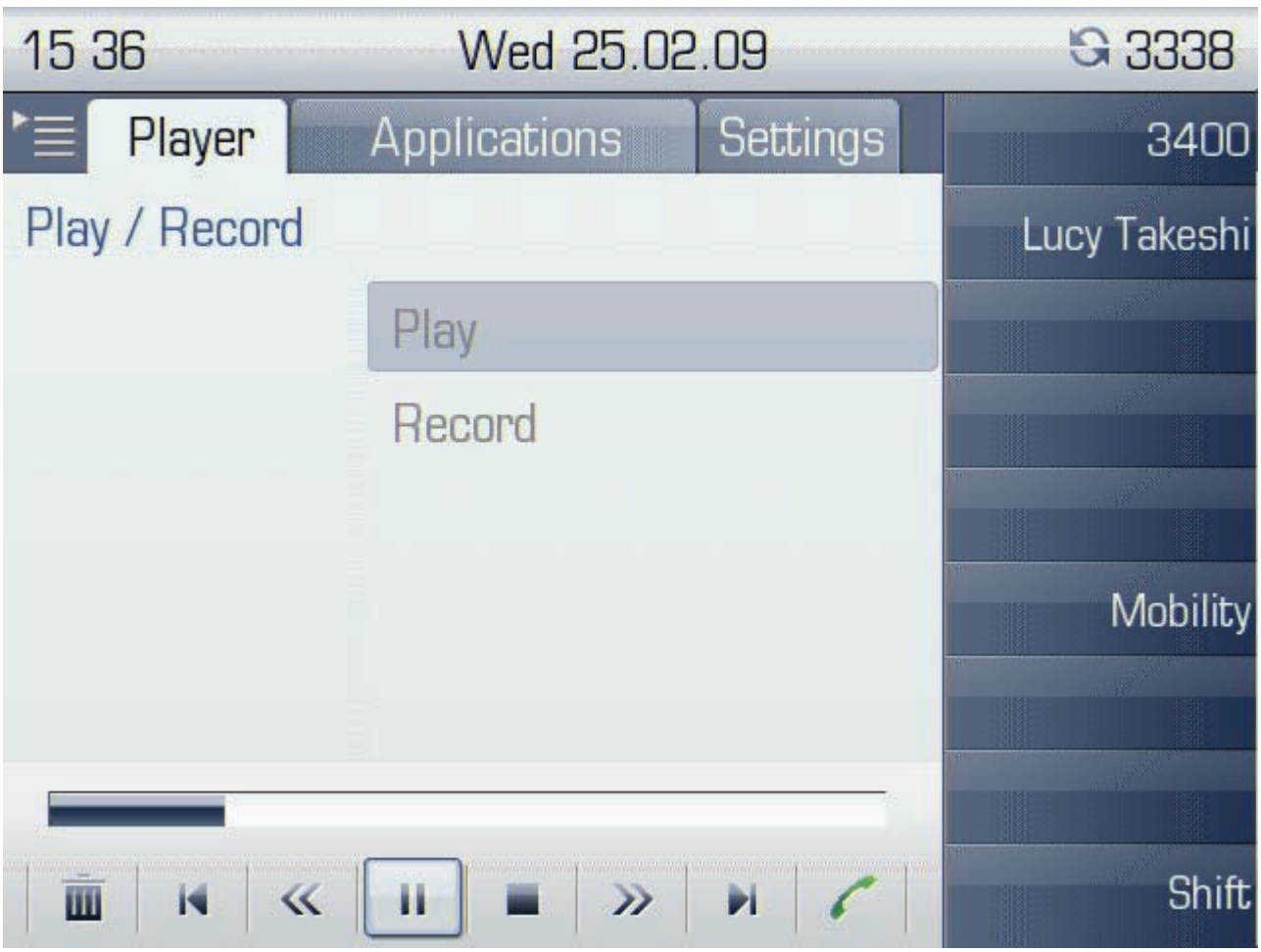
  The `Interactive` attribute must be set to `USER` or `AUTO`, according to the current state and mode of the Player. For details, please refer to Table 4-6.

Example

```
<IppPhone>
   <IppDisplay InitialScreen="1">
      <IppScreen ID="1" HiddenCount="0" CommandCount="2">
         <IppForm ItemCount="2">
            <Title>Play / Record</Title>
            <Url>http://subdomain.domain/path/program</Url>
             <IppItem CommandCount="1">
                 <IppStringItem>
                    <Label></Label>
                    <Text>Play</Text>
                  </IppStringItem>
                   <IppCommand Type="SCREEN" DisplayOn="LISTITEM"
Priority="0">
                      <Label>Select</Label>
                       <ScreenID>2</ScreenID>
                   </IppCommand>
             </IppItem>
             <IppItem CommandCount="1">
                 <IppStringItem>
                     <Label></Label>
                       <Text>Record</Text>
                   </IppStringItem>
                  <IppCommand Type="SCREEN" DisplayOn="LISTITEM" Pri-
ority="0">
                    <Label>Select</Label>
                    <ScreenID>3</ScreenID>
                 </IppCommand>
             </IppItem>
         </IppForm>
      </IppScreen>
      <IppScreen ID="2" HiddenCount="1" CommandCount="2">
         <IppPlayer State="PLAYING" Mode="CALL" Key="Myplayer">
             <Url>http://subdomain.domain/path/program</Url>
              <IppGauge Interactive="AUTO" Key="G1">
                  <Label></Label>
                  <Maximum>100</Maximum>
                   <Initial>0</Initial>
              </IppGauge>
         </IppPlayer>
      </IppScreen>
```

```
      <IppScreen ID="3" HiddenCount="1" CommandCount="2">
         <IppPlayer State="RECORDING" Mode="RECORD" Key="Myplayer">
            <Url>http://subdomain.domain/path/program</Url>
            <IppGauge Interactive="AUTO" Key="G1">
               <Label>Interactive Gauge</Label>
               <Maximum>100</Maximum>
               <Initial>0</Initial>
            </IppGauge>
         </IppPlayer>
      </IppScreen>
   </IppDisplay>
</IppPhone>
```

Screenshot



4.16 Phone Number

This element enables the translation of a given phone number which is stored in the local phone book. The phone number can be translated to:

- the name of the participant,
- the picture clip associated to the participant, if available,
- or, alternatively, an icon of the phone type (e.g. mobile number, home number, etc.).

If the translation fails, alternative text can be displayed instead of the phone number.

Depending on the GUI element wherein the Phone Number is placed, some or all of the information listed above may be displayed instead of or in addition to the phone number.

For example,

- An Alert can display: the name, number, and picture clip of the participant (if available) or an icon of the phone type (see Section 4.8, "Alert").

  In alerts, if the name and number are both displayed, the name is placed on the top line, and the number is placed immediately below it.

- A single column List or the left hand column in a multiple column List can display a picture or an icon followed by the name or number (see Section 4.9, "List").
- Any other column in a multiple column List can only display the name or number (see Section 4.9, "List").

The following attributes are available to the `IppPhoneNumber` element:

Attribute	Value	Function/Remarks
`ImageType` (optional)	`PHONETYPE`	The icon of the phone type according to the given phone number is displayed.
	`PICTURECLIP`	The picture clip associated to the participant is displayed, if available.
`NumberType` (optional)	`NUMBER`	Only the number is displayed.
	`NAME`	Only the corresponding name in the Phone Book is displayed.
	`BOTH`	Both the number and the corresponding name in the Phone Book are displayed

For consistency with the style for presentations of call identities on OpenStage, `ImageType` should equal `PICTURECLIP` and `NumberType` should equal `NAME`.

Syntax

```
<IppPhoneNumber ImageType="[PHONETYPE|PICTURECLIP]"
NumberType="[NUMBER|NAME|BOTH]">
   <AltText> </AltText>
</IppPhoneNumber>
```

Child elements

- `AltText`

  In case the number does not exist in the Phone Book, the alternative text is displayed instead of the number, name or picture clip.

5 Push Capability

A Push request is a specific HTTP or HTTPS (future) request that is sent from the remote server to the Web service of the telephone. For further information, see Section 5.1, "The 'Push' Request".

The server must communicate the nature of the unsolicited 'Push' request. There are four different types of 'Push'. The type influences when, where, and how the information is displayed to the user.

The four different types are:

1. **Active**: The request contains non-essential information which should only be displayed if the required XML application is already running. The information is displayed in a non-intrusive manner. If the required XML application is not running, nothing changes, as it is not started on Push.
2. **Queue**: The request contains information that does not have to be presented immediately to the user (e.g. offers, restaurant menu, message of the day). If the required XML application is not running, it is started on Push. The user chooses when to display the information.
3. **Indicate**: The request contains non-essential information (e.g. received new email, update message playback progress). If the required XML application is not running, it is started on Push. The information is displayed to the user in a non-intrusive manner. The user can choose to continue with the current application or switch to the new information.

For most cases, Indicate is the best option. It is a compromise between displaying most information and limiting the interruption to the currently displayed application.

4. **Force**: The request contains urgent information that must be immediately displayed to the user. The user is automatically switched from the current application (telephony or XML) to the required XML application.

   This option should only be used when the XML application requires the user's attention to inform them of something or request their input. This option is disruptive to the user if they are using another application.

The following table shows which actions are performed by which push request type, considering the possible sub-conditions:

Action	**sub-condition**		**'Push' Type**		
		Active	**Queue**	**Indicate**	**Force**
XML Application started on 'Push'	**Program is configured and not running**	No	Yes	Yes	Yes
	Program is not configured	Error case - The Push request is ignored			
XML document loaded from the nominated URL on the server		Yes, if program is already running.	Yes	Yes	Yes

Table 5-1

Action	sub-condition		'Push' Type		
		Active	**Queue**	**Indicate**	**Force**
XML document rendered	**XML document contains IppList, IppTextBox, or IppForm**	Yes, appropriate tab updated if program is already running.	Yes, appropriate tab updated.	Yes, appropriate tab updated.	Yes, appropriate tab updated.
	XML document contains IppAlert	Popup is only created and displayed if program is already running.	No popup is created or displayed.	Popup is created and displayed.	Popup is created and displayed.
	XML document contains IppPlayer	Player is only created and displayed if program is already running.	No player is created or displayed.	Player is created and displayed.	Player is created and displayed.
XML Application gains screen focus		No	No	No	Yes, XML tab is displayed.
Popup (or Player) shown to User		Yes, if no other popup or prompt is showing.	No	Yes, if no other popup or prompt is showing.	Yes, but not while a prompt (e.g. incoming call Alert or Dialer) is showing.
Popup (or Player) gains input focus		Yes, if shown.	No	Yes, if shown.	Yes
Screensaver		Stopped for popup or player.	No change	Stopped	Stopped
Phone locked		No change	No change	No change	No change

Table 5-1

5.1 The 'Push' Request

The Push request is an HTTP or HTTPS (future) request sent to the following phone-side URL:

`http://<ip-address>:<port-number>/server_push.html/ServerPush`

Example:

`http://137.223.234.11:8085/server_push.html/ServerPush`

The Push request must contain the following Information:

Key	Value	Function
`ServerAddr`	IP address	IP address of the server where the program is located.
`ServerPort`	Port numbe*r*	Port number of the server where the program is located.
`ProgramName`	Resource name	Entity part of the URI pointing to the server-side program. For instance, this can be an XML file name, or the name of a servlet. To start a tab on demand, this parameter must contain the appropriate application name (**Tab 1...3 Application name**, see Section 3.1, "Configuring an Application on the Phone" and Section 3.4, "Application Suite Using Tabs On Demand").

Table 5-2

Key	Value	Function
`RequestType`	`active`	The XML document is loaded, if the program is already started. If the document contains a popup (`IppAlert`) or a player (`IppPlayer`), and no other popup or player is shown already, the player or popup is displayed. For detailed information, see the table in Section 5, "Active".
	`queue`	The program is started in the background and the XML document is loaded. For detailed information, see the table in Section 5, "Queue".
	`indicate`	The program is started and the XML document is loaded. If the document contains a popup (`IppAlert`) or a player (`IppPlayer`), and no other popup or player is shown already, the player or popup is displayed. For detailed information, see the table in Section 5, "Indicate".
	`force`	The program is started and the XML document is loaded. It is displayed to the user with input focus, unless a prompt (e.g. incoming call Alert or Dialer) is showing. For detailed information, see the table in Section 5, "Force".
`MidletName`	Name of the MIDlet	Name of the application that is to be controlled by the Push request. Depending on the type of Push request, and the current state of the application, the application may be started by the Push request. This parameter is equivalent to the **Application name** entered when the application was configured on the telephone (see Section 3.1, "Configuring an Application on the Phone").
`ServerProtocol`	Communication protocol	Indicates the protocol use by the server for communicating to the phone. Examples: `http`, `https` (future).
`ServerContextKey`	Key	The key for the key/value pair that is returned to the program running on the server. The server uses this key/value pair to detect whether the XML document request goes back to the previous Push request that had been sent to the telephone.
`ServerContextValue`	Value	The value for the key/value pair that is returned to the program running on the server.

Table 5-2

6 Appendix

6.1 Glossary

Term/Abbreviation	Definition/Description
HFA	HiPath Feature Access
jad	Java Application Descriptor
Jar	Java Archive File
SIP	Session Initiation Protocol
Web Based Manager	Telephone configuration program that is used via a web browser

6.2 XML Schema

This appendix contains the entire XML schema supported by the HFA and SIP variants of the OpenStage 60/80.
You can download the current version of the XML schema here:
http://wiki.siemens-enterprise.com/index.php/OpenStage_XML_Applications#Resources

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

   <xs:annotation>
      <xs:documentation xml:lang="en">
         XML Schema for Siemens IP Phone Generic MIDlet. Version 5.0
         Copyright 2007 Siemens Enterprise Communications Limited. All
         rights reserved.
      </xs:documentation>
   </xs:annotation>

   <xs:element name="IppPhone">
      <xs:complexType>
         <xs:sequence>
            <xs:choice>
               <xs:element ref="IppDisplay"/>
            </xs:choice>
         </xs:sequence>
      </xs:complexType>
   </xs:element>

   <xs:element name="IppDisplay">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="IppScreen" maxOccurs="5"/>
         </xs:sequence>
         <xs:attribute name="InitialScreen" type="xs:nonNegativeInte-
ger" use="optional"/>
         <xs:attribute name="UpdateScreen" type="xs:nonNegativeInte-
ger" use="optional"/>
      </xs:complexType>
   </xs:element>
```

```
   <xs:element name="IppScreen">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="IppKey" minOccurs="0"/>
            <xs:choice>
               <xs:element ref="IppAlert" minOccurs="0"/>
               <xs:element ref="IppList" minOccurs="0"/>
               <xs:element ref="IppTextBox" minOccurs="0"/>
               <xs:element ref="IppForm" minOccurs="0"/>
               <xs:element ref="IppPlayer" minOccurs="0"/>
            </xs:choice>
            <xs:element ref="IppAction" minOccurs="0"/>
             <xs:element ref="IppCommand" minOccurs="0" maxOc-
curs="unbounded"/>
              <xs:element ref="IppHidden" minOccurs="0" maxOccurs="un-
bounded"/>
            <xs:element ref="IppTicker" minOccurs="0"/>
         </xs:sequence>
         <xs:attribute name="HiddenCount" type="xs:nonNegativeInte-
ger" use="required"/>
          <xs:attribute name="CommandCount" type="xs:nonNegativeInte-
ger" use="required"/>
          <xs:attribute name="ID" type="xs:nonNegativeInteger"
use="optional" default="0"/>
          <xs:attribute name="Sound" type="xs:string" use="optional"/
>
         <!-- default="0"/> -->
         <!-- default="0"/> -->
      </xs:complexType>
   </xs:element>
```

```
<xs:element name="IppAction">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="Number" minOccurs="0">
            <xs:simpleType>
               <xs:restriction base="xs:string">
                  <xs:pattern value="\d*"/>
               </xs:restriction>
            </xs:simpleType>
         </xs:element>
      </xs:sequence>
      <xs:attribute name="Type" use="required">
         <xs:simpleType>
            <xs:restriction base="xs:string">
               <xs:enumeration value="MAKECALL"/>
               <xs:enumeration value="ENDCALL"/>
               <xs:enumeration value="TURNLEDON"/>
               <xs:enumeration value="TURNLEDOFF"/>
            </xs:restriction>
         </xs:simpleType>
      </xs:attribute>
   </xs:complexType>
</xs:element>
```

```
<xs:element name="IppAlert">
   <xs:complexType>
      <xs:sequence>
         <xs:element ref="Title" minOccurs="0"/>
         <xs:element ref="Url" minOccurs="0"/>
         <xs:element ref="Text" minOccurs="0"/>
         <xs:element ref="Image" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute name="Type" use="optional" default="ERROR">
         <xs:simpleType>
            <xs:restriction base="xs:string">
               <xs:enumeration value="ALARM"/>
               <xs:enumeration value="CONFIRMATION"/>
               <xs:enumeration value="ERROR"/>
               <xs:enumeration value="INFO"/>
               <xs:enumeration value="WARNING"/>
            </xs:restriction>
         </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="Delay" use="optional" default="FOREVER">
         <xs:simpleType>
            <xs:union memberTypes="xs:positiveInteger">
               <xs:simpleType>
                  <xs:restriction base="xs:string">
                     <xs:enumeration value="FOREVER"/>
                  </xs:restriction>
               </xs:simpleType>
            </xs:union>
         </xs:simpleType>
      </xs:attribute>
   </xs:complexType>
</xs:element>
```

```
   <xs:element name="IppButton">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="Label" minOccurs="0"/>
            <xs:element ref="Image" minOccurs="0"/>
         </xs:sequence>
         <xs:attribute name="Layout" use="optional" default="DE-
FAULT">
            <xs:simpleType>
               <xs:restriction base="xs:string">
                  <xs:enumeration value="CENTER"/>
                  <xs:enumeration value="DEFAULT"/>
                  <xs:enumeration value="LEFT"/>
                  <xs:enumeration value="NEWLINE_AFTER"/>
                  <xs:enumeration value="NEWLINE_BEFORE"/>
                  <xs:enumeration value="RIGHT"/>
               </xs:restriction>
            </xs:simpleType>
         </xs:attribute>
         <xs:attribute name="Type" use="optional" default="IMAGE">
            <xs:simpleType>
               <xs:restriction base="xs:string">
                  <xs:enumeration value="IMAGE"/>
               </xs:restriction>
            </xs:simpleType>
         </xs:attribute>
         <xs:attribute ref="Key" use="required"/>
         <xs:attribute ref="Value" use="required"/>
      </xs:complexType>
   </xs:element>
```

```
   <xs:element name="IppChoiceGroup">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="Label" minOccurs="0"/>
            <xs:element ref="Option" minOccurs="0" maxOccurs="un-
bounded"/>
         </xs:sequence>
         <xs:attribute name="Numbered" default="NO">
            <xs:simpleType>
               <xs:restriction base="xs:string">
                  <xs:enumeration value="NO"/>
                  <xs:enumeration value="YES"/>
               </xs:restriction>
            </xs:simpleType>
         </xs:attribute>
         <xs:attribute name="Type" use="required">
            <!-- default="EXCLUSIVE"> -->
            <xs:simpleType>
               <xs:restriction base="xs:string">
                  <xs:enumeration value="EXCLUSIVE"/>
                     <xs:enumeration value="MULTIPLE"/>
                     <xs:enumeration value="POPUP"/>
               </xs:restriction>
            </xs:simpleType>
         </xs:attribute>
         <xs:attribute name="Count" type="xs:nonNegativeInteger"
use="required"/>
            <!-- default="0"/> -->
      </xs:complexType>
   </xs:element>
```

```
   <xs:element name="IppCommand">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="Label" type="xs:string" minOccurs="0"/>
            <xs:element name="ScreenID" type="xs:nonNegativeInteger"
minOccurs="0"/>
         </xs:sequence>
         <xs:attribute name="Type" use="required">
            <xs:simpleType>
               <xs:restriction base="xs:string">
                  <xs:enumeration value="SELECT"/>
                  <xs:enumeration value="BACK"/>
                  <xs:enumeration value="UPDATE"/>
                  <xs:enumeration value="SCREEN"/>
                  <xs:enumeration value="CANCEL"/>
                  <xs:enumeration value="EXIT"/>
               </xs:restriction>
            </xs:simpleType>
         </xs:attribute>
         <xs:attribute name="Priority" use="optional" default="0">
            <xs:simpleType>
               <xs:restriction base="xs:nonNegativeInteger">
                  <xs:maxInclusive value="9"/>
               </xs:restriction>
            </xs:simpleType>
         </xs:attribute>
         <xs:attribute name="Auto" use="optional" default="0">
            <xs:simpleType>
               <xs:restriction base="xs:nonNegativeInteger"/>
            </xs:simpleType>
         </xs:attribute>
         <xs:attribute name="DisplayOn" use="optional" default="OP-
TIONS">
            <xs:simpleType>
               <xs:restriction base="xs:string">
                  <xs:enumeration value="OPTIONS"/>
                  <xs:enumeration value="LISTITEM"/>
                  <xs:enumeration value="BOTH"/>
               </xs:restriction>
            </xs:simpleType>
         </xs:attribute>
         <xs:attribute name="Select" use="optional" default="NO">
            <xs:simpleType>
               <xs:restriction base="xs:string">
```

```
                <xs:enumeration value="YES"/>
                <xs:enumeration value="NO"/>
             </xs:restriction>
          </xs:simpleType>
       </xs:attribute>
       <xs:attribute name="Default" use="optional" default="NO">
          <xs:simpleType>
             <xs:restriction base="xs:string">
                <xs:enumeration value="YES"/>
                <xs:enumeration value="NO"/>
             </xs:restriction>
          </xs:simpleType>
       </xs:attribute>
       <xs:attribute ref="Key" use="optional"/>
       <xs:attribute ref="Value" use="optional"/>
    </xs:complexType>
</xs:element>
```

```
<xs:element name="IppDateField">
   <xs:complexType>
      <xs:sequence>
         <xs:element ref="Label" minOccurs="0"/>
         <xs:element ref="Timezone" minOccurs="0"/>
         <xs:element ref="Date" minOccurs="0"/>
         <xs:element ref="Time" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute name="Mode" use="optional" default="DATETIME">
         <xs:simpleType>
            <xs:restriction base="xs:string">
               <xs:enumeration value="DATE"/>
               <xs:enumeration value="TIME"/>
               <xs:enumeration value="DATETIME"/>
            </xs:restriction>
         </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="Default" use="optional" default="NULL">
         <xs:simpleType>
            <xs:restriction base="xs:string">
               <xs:enumeration value="NULL"/>
               <xs:enumeration value="MODE"/>
            </xs:restriction>
         </xs:simpleType>
      </xs:attribute>
      <xs:attribute ref="DateKey" use="optional"/>
      <xs:attribute ref="TimeKey" use="optional"/>
   </xs:complexType>
</xs:element>
```

```
   <xs:element name="IppForm">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="Title" minOccurs="0"/>
            <xs:element ref="Url" minOccurs="0"/>
            <!-- IppForm may contain any combination of none, one, or
            more IppButton, IppChoiceGroup, IppDateField, IppGauge,
            IppImageItem, IppItem, IppSpacer, IppStringItem, and
            Ipp TextField in any order. The correct XML syntax should
            surround each one by "<IppItem></IppItem>". However, this
            increases the size of the XML document and has an impact
            on performance.
            When validating a simple IppForm with only one type of
            element, each element does not have to be surrounded by
            <IppItem></IppItem>. When validating a complex IppForm
            with many different types of elements, each elements must
            be surrounded by <IppItem></IppItem>. When the XML
            document for a complex IppForm has been validated, the
            "IppItem" tags can be removed from the XML document. If
            they remain in the XML document, they will be correctly
            parsed.-->

            <xs:choice>
               <xs:element ref="IppButton" minOccurs="0" maxOc-
curs="unbounded"/>
               <xs:element ref="IppChoiceGroup" minOccurs="0" maxOc-
curs="unbounded"/>
               <xs:element ref="IppDateField" minOccurs="0" maxOc-
curs="unbounded"/>
               <xs:element ref="IppGauge" minOccurs="0" maxOc-
curs="unbounded"/>
               <xs:element ref="IppImageItem" minOccurs="0" maxOc-
curs="unbounded"/>
               <xs:element ref="IppItem" minOccurs="0" maxOccurs="un-
bounded"/>
               <xs:element ref="IppSpacer" minOccurs="0" maxOc-
curs="unbounded"/>
               <xs:element ref="IppStringItem" minOccurs="0" maxOc-
curs="unbounded"/>
               <xs:element ref="IppTextField" minOccurs="0" maxOc-
curs="unbounded"/>
            </xs:choice>
         </xs:sequence>
         <xs:attribute name="ItemCount" type="xs:nonNegativeInteger"
use="required"/>
```

```
         <xs:attribute name="Proportion" use="optional" de-
fault="40_60">
            <xs:simpleType>
               <xs:restriction base="xs:string">
                  <xs:enumeration value="40_60"/>
                  <xs:enumeration value="50_50"/>
                  <xs:enumeration value="60_40"/>
                  <xs:enumeration value="75_25"/>
                  <xs:enumeration value="25_75"/>
                  <xs:enumeration value="15_85"/>
                  <xs:enumeration value="0_100"/>
               </xs:restriction>
            </xs:simpleType>
         </xs:attribute>
         <!-- default="0"/> -->
      </xs:complexType>
   </xs:element>
```

```
   <xs:element name="IppGauge">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="Label" minOccurs="0"/>
            <xs:element ref="Maximum" minOccurs="0"/>
            <xs:element ref="Initial" minOccurs="0"/>
         </xs:sequence>
         <xs:attribute name="Interactive" use="optional" default="US-
ER">
            <xs:simpleType>
               <xs:restriction base="xs:string">
                  <xs:enumeration value="USER"/>
                  <xs:enumeration value="AUTO"/>
               </xs:restriction>
            </xs:simpleType>
         </xs:attribute>
         <xs:attribute ref="Key" use="optional"/>
      </xs:complexType>
   </xs:element>

   <xs:element name="IppHidden">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="Value" minOccurs="0"/>
         </xs:sequence>
         <xs:attribute name="Type" use="required">
         <!-- default="VALUE"> -->
            <xs:simpleType>
               <xs:restriction base="xs:string">
                  <xs:enumeration value="VALUE"/>
                  <xs:enumeration value="PHONENUMBER"/>
                  <xs:enumeration value="IPADDRESS"/>
               </xs:restriction>
            </xs:simpleType>
         </xs:attribute>
         <xs:attribute ref="Key" use="required"/>
      </xs:complexType>
   </xs:element>
```

```
   <xs:element name="IppImageItem">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="Label" minOccurs="0"/>
            <xs:element ref="Image" minOccurs="0"/>
            <xs:element ref="AltText" minOccurs="0"/>
         </xs:sequence>
         <xs:attribute name="Layout" use="optional" default="CENTER">
            <xs:simpleType>
               <xs:restriction base="xs:string">
                  <xs:enumeration value="CENTER"/>
                  <xs:enumeration value="DEFAULT"/>
                  <xs:enumeration value="LEFT"/>
                  <xs:enumeration value="NEWLINE_AFTER"/>
                  <xs:enumeration value="NEWLINE_BEFORE"/>
                  <xs:enumeration value="RIGHT"/>
               </xs:restriction>
            </xs:simpleType>
         </xs:attribute>
      </xs:complexType>
   </xs:element>

   <xs:element name="IppItem">
      <xs:complexType>
         <xs:sequence>
            <xs:choice>
               <xs:element ref="IppButton" minOccurs="0"/>
               <xs:element ref="IppChoiceGroup" minOccurs="0"/>
               <xs:element ref="IppDateField" minOccurs="0"/>
               <xs:element ref="IppGauge" minOccurs="0"/>
               <xs:element ref="IppImageItem" minOccurs="0"/>
               <xs:element ref="IppSpacer" minOccurs="0"/>
               <xs:element ref="IppStringItem" minOccurs="0"/>
               <xs:element ref="IppTextField" minOccurs="0"/>
            </xs:choice>
            <xs:element ref="IppCommand" minOccurs="0" maxOccurs="un-
bounded"/>
         </xs:sequence>
         <xs:attribute name="CommandCount" type="xs:nonNegativeInte-
ger" use="required"/>
         <!-- default="0"/> -->
      </xs:complexType>
   </xs:element>
```

```
   <xs:element name="IppKey">
      <xs:complexType>
         <xs:attribute name="Keypad" use="required">
            <!--default="NO"> -->
            <xs:simpleType>
               <xs:restriction base="xs:string">
                  <xs:enumeration value="NO"/>
                  <xs:enumeration value="YES"/>
               </xs:restriction>
            </xs:simpleType>
         </xs:attribute>
         <xs:attribute name="SendKeys" use="required">
            <!--default="NO"> -->
            <xs:simpleType>
               <xs:restriction base="xs:string">
                  <xs:enumeration value="NO"/>
                  <xs:enumeration value="YES"/>
               </xs:restriction>
            </xs:simpleType>
         </xs:attribute>
         <xs:attribute name="BufferKeys" use="optional" default="NO">
            <xs:simpleType>
               <xs:restriction base="xs:string">
                  <xs:enumeration value="NO"/>
                  <xs:enumeration value="YES"/>
                  <xs:enumeration value="SUBSEQUENT"/>
               </xs:restriction>
            </xs:simpleType>
         </xs:attribute>
         <xs:attribute name="BufferLength" use="optional" de-
fault="0">
            <xs:simpleType>
               <xs:restriction base="xs:nonNegativeInteger"/>
            </xs:simpleType>
         </xs:attribute>
         <xs:attribute name="TermKey" use="optional" default=" ">
            <xs:simpleType>
               <xs:restriction base="xs:string">
                  <xs:pattern value="[ *#0-9]{1}"/>
               </xs:restriction>
            </xs:simpleType>
         </xs:attribute>
```

```
         <xs:attribute ref="UrlKey" use="required"/>
      </xs:complexType>
   </xs:element>

   <xs:element name="IppList">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="Title" minOccurs="0"/>
            <xs:element ref="Url" minOccurs="0"/>
            <xs:element ref="Option" minOccurs="0" maxOccurs="un-
bounded"/>
         </xs:sequence>
         <xs:attribute name="Numbered" use="optional" default="NO">
            <xs:simpleType>
               <xs:restriction base="xs:string">
                  <xs:enumeration value="NO"/>
                  <xs:enumeration value="YES"/>
               </xs:restriction>
            </xs:simpleType>
         </xs:attribute>
         <xs:attribute name="Type" use="required">
            <!-- default="IMPLICIT"> -->
            <xs:simpleType>
               <xs:restriction base="xs:string">
                  <xs:enumeration value="IMPLICIT"/>
                  <xs:enumeration value="EXCLUSIVE"/>
                  <xs:enumeration value="MULTIPLE"/>
               </xs:restriction>
            </xs:simpleType>
         </xs:attribute>
         <xs:attribute name="Count" type="xs:nonNegativeInteger"
use="required"/>
         <xs:attribute name="Columns" use="optional" default="1">
            <xs:simpleType>
               <xs:restriction base="xs:nonNegativeInteger">
                  <xs:enumeration value="1"/>
                  <xs:enumeration value="3"/>
               </xs:restriction>
            </xs:simpleType>
         </xs:attribute>
         <xs:attribute ref="FColor"/>
         <xs:attribute ref="TColor"/>
         <xs:attribute ref="SFColor"/>
         <xs:attribute ref="STColor"/>
```

```
         <!-- default="0"/> -->
      </xs:complexType>
   </xs:element>

   <xs:element name="IppPhoneNumber">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="AltText" minOccurs="0"/>
         </xs:sequence>
         <xs:attribute name="ImageType" use="optional" default="NOIM-
AGE">
            <xs:simpleType>
               <xs:restriction base="xs:string">
                  <xs:enumeration value="NOIMAGE"/>
                  <xs:enumeration value="PHONETYPE"/>
                  <xs:enumeration value="PICTURECLIP"/>
               </xs:restriction>
            </xs:simpleType>
         </xs:attribute>
         <xs:attribute name="NumberType" use="optional" default="NUM-
BER">
            <xs:simpleType>
               <xs:restriction base="xs:string">
                  <xs:enumeration value="NUMBER"/>
                  <xs:enumeration value="NAME"/>
                  <xs:enumeration value="BOTH"/>
               </xs:restriction>
            </xs:simpleType>
         </xs:attribute>
      </xs:complexType>
   </xs:element>
```

```
<xs:element name="IppPlayer">
   <xs:complexType>
      <xs:sequence>
         <xs:element ref="Url" minOccurs="0"/>
         <xs:element ref="IppGauge" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute name="State" use="required">
      <!-- default="STOPPED"> -->
         <xs:simpleType>
            <xs:restriction base="xs:string">
               <xs:enumeration value="PLAYING"/>
               <xs:enumeration value="RECORDING"/>
               <xs:enumeration value="STOPPED"/>
            </xs:restriction>
         </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="Mode" use="required">
         <!-- default="CALL"> -->
         <xs:simpleType>
            <xs:restriction base="xs:string">
               <xs:enumeration value="CALL"/>
               <xs:enumeration value="RECORD"/>
            </xs:restriction>
         </xs:simpleType>
      </xs:attribute>
      <xs:attribute ref="Key" use="required"/>
   </xs:complexType>
</xs:element>
```

```
<xs:element name="IppSpacer">
   <xs:complexType>
      <xs:attribute name="NewLine" use="required">
         <!--- default="NEWLINE_BEF_AFT"> -->
            <xs:simpleType>
               <xs:restriction base="xs:string">
                  <xs:enumeration value="NEWLINE_AFTER"/>
                  <xs:enumeration value="NEWLINE_BEFORE"/>
                  <xs:enumeration value="NEWLINE_BEF_AFT"/>
               </xs:restriction>
            </xs:simpleType>
         </xs:attribute>
         <xs:attribute name="Height" use="optional" default="0">
            <xs:simpleType>
               <xs:restriction base="xs:nonNegativeInteger"/>
            </xs:simpleType>
         </xs:attribute>
         <xs:attribute name="Width" use="optional" default="0">
            <xs:simpleType>
               <xs:restriction base="xs:nonNegativeInteger"/>
                  </xs:simpleType>
         </xs:attribute>
      </xs:complexType>
   </xs:element>
```

```
   <xs:element name="IppStringItem">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="Label" minOccurs="0"/>
            <xs:element ref="Text" minOccurs="0"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>

   <xs:element name="IppTextBox">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="Title" minOccurs="0"/>
            <xs:element ref="Text" minOccurs="0"/>
            <xs:element ref="Url" minOccurs="0"/>
         </xs:sequence>
         <xs:attribute name="MaxSize" type="xs:nonNegativeInteger"
use="optional"/>
         <xs:attribute name="Constraint" use="optional" de-
fault="ANY">
            <xs:simpleType>
               <xs:restriction base="xs:string">
                  <xs:enumeration value="ANY"/>
                  <xs:enumeration value="NUMERIC"/>
                  <xs:enumeration value="PASSWORD"/>
                  <xs:enumeration value="PHONENUMBER"/>
                  <xs:enumeration value="URL"/>
                  <xs:enumeration value="EMAILADDR"/>
               </xs:restriction>
            </xs:simpleType>
         </xs:attribute>
         <xs:attribute name="Password" use="optional" default="NO">
            <xs:simpleType>
               <xs:restriction base="xs:string">
                  <xs:enumeration value="YES"/>
                  <xs:enumeration value="NO"/>
               </xs:restriction>
            </xs:simpleType>
         </xs:attribute>
         <xs:attribute name="Uneditable" use="optional" default="NO">
            <xs:simpleType>
               <xs:restriction base="xs:string">
                  <xs:enumeration value="YES"/>
                  <xs:enumeration value="NO"/>
               </xs:restriction>
```

```
            </xs:simpleType>
         </xs:attribute>
         <xs:attribute ref="Default" use="optional" default="NULL"/>
         <xs:attribute ref="Key" use="optional"/>
      </xs:complexType>
   </xs:element>
```

```
   <xs:element name="IppTextField">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="Label" minOccurs="0"/>
            <xs:element ref="Text" minOccurs="0"/>
         </xs:sequence>
         <xs:attribute name="MaxSize" type="xs:nonNegativeInteger"
use="optional"/>
         <xs:attribute name="Constraint" use="optional" de-
fault="ANY">
            <xs:simpleType>
               <xs:restriction base="xs:string">
                  <xs:enumeration value="ANY"/>
                  <xs:enumeration value="NUMERIC"/>
                  <xs:enumeration value="PASSWORD"/>
                  <xs:enumeration value="PHONENUMBER"/>
                  <xs:enumeration value="URL"/>
                  <xs:enumeration value="EMAILADDR"/>
                  <xs:enumeration value="VALIDATE"/>
               </xs:restriction>
            </xs:simpleType>
         </xs:attribute>
         <xs:attribute name="Password" use="optional" default="NO">
            <xs:simpleType>
               <xs:restriction base="xs:string">
                  <xs:enumeration value="YES"/>
                  <xs:enumeration value="NO"/>
               </xs:restriction>
            </xs:simpleType>
         </xs:attribute>
         <xs:attribute name="Uneditable" use="optional" default="NO">
            <xs:simpleType>
               <xs:restriction base="xs:string">
                  <xs:enumeration value="YES"/>
                  <xs:enumeration value="NO"/>
               </xs:restriction>
            </xs:simpleType>
         </xs:attribute>
         <xs:attribute ref="Default" use="optional" default="NULL"/>
         <xs:attribute ref="Key" use="optional"/>
      </xs:complexType>
   </xs:element>
```

```
    <xs:element name="IppTicker">
       <xs:complexType>
          <xs:sequence>
             <xs:element ref="Text" minOccurs="0"/>
          </xs:sequence>
       </xs:complexType>
    </xs:element>

    <xs:element name="AltText" type="xs:string"/>

    <xs:element name="Date">
       <xs:simpleType>
          <xs:restriction base="xs:string">
             <xs:pattern value="\d{4}-\d{2}-\d{2}"/>
          </xs:restriction>
       </xs:simpleType>
    </xs:element>

    <xs:element name="Image">
       <xs:complexType>
          <xs:simpleContent>
             <xs:extension base="imagePath">
                <xs:attribute name="Cache" type="xs:string"/>
             </xs:extension>
          </xs:simpleContent>
       </xs:complexType>
    </xs:element>

<xs:element name="Initial" type="gaugeTimeType"/>

<xs:element name="Label" type="xs:string"/>

<xs:element name="Maximum" type="gaugeTimeType"/>
```

```
   <xs:element name="Option">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="OptionText" maxOccurs="3"/>
            <xs:element ref="Image" minOccurs="0"/>
         </xs:sequence>
         <xs:attribute name="ID" type="xs:nonNegativeInteger"
use="optional"/>
         <xs:attribute name="Selected" use="required">
            <!-- default="FALSE"> -->
            <xs:simpleType>
               <xs:restriction base="xs:string">
                  <xs:enumeration value="TRUE"/>
                  <xs:enumeration value="FALSE"/>
               </xs:restriction>
            </xs:simpleType>
         </xs:attribute>
         <xs:attribute ref="Key" use="required"/>
         <xs:attribute ref="Value" use="required"/>
         <xs:attribute ref="FColor"/>
         <xs:attribute ref="TColor"/>
         <xs:attribute ref="SFColor"/>
         <xs:attribute ref="STColor"/>
      </xs:complexType>
   </xs:element>
```

```
<xs:element name="OptionText">
   <xs:complexType mixed="true">
      <xs:sequence>
         <xs:element ref="IppPhoneNumber" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute name="State" default="NORMAL">
         <xs:simpleType>
            <xs:restriction base="xs:string">
               <xs:enumeration value="NORMAL"/>
               <xs:enumeration value="GRAY"/>
            </xs:restriction>
         </xs:simpleType>
      </xs:attribute>
   </xs:complexType>
</xs:element>

<xs:element name="Text">
   <xs:complexType mixed="true">
      <xs:sequence>
         <xs:element ref="IppPhoneNumber" minOccurs="0"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>

<xs:element name="Time">
   <xs:simpleType>
      <xs:restriction base="xs:string">
         <xs:pattern value="\d{2}:\d{2}:\d{2}.\d{3}"/>
      </xs:restriction>
   </xs:simpleType>
</xs:element>

<xs:element name="Timezone" type="timeZoneType"/>

<xs:element name="Title" type="xs:string"/>

<xs:element name="Url" type="xs:anyURI"/>
```

```
<xs:element name="Value" type="xs:string"/>

<xs:attribute name="DateKey" type="xs:string"/>

<xs:attribute name="Default">
    <xs:simpleType>
       <xs:restriction base="xs:string">
          <xs:enumeration value="NULL"/>
          <xs:enumeration value="TEXT"/>
          <xs:enumeration value="PHONENUMBER"/>
       </xs:restriction>
    </xs:simpleType>
</xs:attribute>

<xs:attribute name="FColor" type="colourType"/>

<xs:attribute name="Key" type="xs:string"/>

<xs:attribute name="SFColor" type="colourType"/>

<xs:attribute name="STColor" type="colourType"/>

<xs:attribute name="TColor" type="colourType"/>

<xs:attribute name="TimeKey" type="xs:string"/>

<xs:attribute name="Value" type="xs:string"/>

<xs:attribute name="UrlKey" type="xs:string"/>
```

```
<xs:simpleType name="colourType">
   <xs:union>
      <xs:simpleType>
         <xs:restriction base="xs:string">
            <xs:pattern value="#[0-9ABCDEF]{6}"/>
         </xs:restriction>
      </xs:simpleType>
      <xs:simpleType>
         <xs:restriction base="xs:string">
            <xs:enumeration value="BLACK"/>
            <xs:enumeration value="RED"/>
            <xs:enumeration value="GREEN"/>
            <xs:enumeration value="BROWN"/>
            <xs:enumeration value="BLUE"/>
            <xs:enumeration value="MAGENTA"/>
            <xs:enumeration value="CYAN"/>
            <xs:enumeration value="LIGHTGRAY"/>
            <xs:enumeration value="DARKGRAY"/>
            <xs:enumeration value="LIGHTRED"/>
            <xs:enumeration value="LIGHTGREEN"/>
            <xs:enumeration value="YELLOW"/>
            <xs:enumeration value="LIGHTBLUE"/>
            <xs:enumeration value="LIGHTMAGENTA"/>
            <xs:enumeration value="LIGHTCYAN"/>
            <xs:enumeration value="WHITE"/>
         </xs:restriction>
      </xs:simpleType>
   </xs:union>
</xs:simpleType>

<xs:simpleType name="gaugeTimeType">
   <xs:union memberTypes="xs:nonNegativeInteger">
      <xs:simpleType>
         <xs:restriction base="xs:string">
            <xs:pattern value="\d{2}:\d{2}"/>
         </xs:restriction>
      </xs:simpleType>
   </xs:union>
</xs:simpleType>
```

```
   <xs:simpleType name="imagePath">
      <xs:union>
         <xs:simpleType>
            <xs:restriction base="xs:string">
               <xs:pattern value=".+"/>
            </xs:restriction>
         </xs:simpleType>
         <xs:simpleType>
            <xs:restriction base="xs:string">
               <xs:pattern value="file:.+"/>
            </xs:restriction>
         </xs:simpleType>
         <xs:simpleType>
            <xs:restriction base="xs:string">
               <xs:pattern value="http.+"/>
            </xs:restriction>
         </xs:simpleType>
         <xs:simpleType>
            <xs:restriction base="xs:string">
               <xs:pattern value="map:.+"/>
            </xs:restriction>
         </xs:simpleType>
      </xs:union>
   </xs:simpleType>

   <xs:simpleType name="timeZoneType">
      <xs:union memberTypes="xs:string">
         <xs:simpleType>
            <xs:restriction base="xs:string">
               <xs:pattern value="GMT[\+|\-]\d*"/>
            </xs:restriction>
         </xs:simpleType>
      </xs:union>
   </xs:simpleType>

</xs:schema>
```